Universidade de Aveiro
2009

Departamento de
Electrónica,Telecomunicações e Informática

Rui Manuel da Cunha Sacadura Botte

Registo Médico Electrónico Transportável - pEHR

Universidade de Aveiro
2009

Departamento de
Electrónica,Telecomunicações e Informática

Rui Manuel da Cunha Sacadura Botte

Registo Médico Electrónico Transportável - pEHR

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Prof. Doutor Carlos Manuel Azevedo Costa, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Dedico este trabalho ao meu pai Elmano, à minha mãe Lúcia e à memória da minha Avó Raquel.

o júri

presidente

Prof. Dr. Tomás Oliveira e Silva
Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Prof. Dr. Ricardo João Cruz Correia
Professor Auxiliar do Departamento de Bioestatística e Informática Médica da Faculdade de Medicina da Universidade do Porto

Prof. Dr. Carlos Manuel Azevedo Costa
Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

agradecimentos

A realização deste trabalho não seria possível apenas com o esforço individual. Para a realização do mesmo contei com ajuda e apoio de algumas pessoas, às quais gostaria de deixar os meus agradecimentos.

Em primeiro lugar ao meu orientador, Prof. Dr. Carlos Costa pela forma como me acompanhou e orientou, pelos conselhos, encorajamento e motivação sem os quais nunca teria sido possível concluir esta dissertação.

Ao Eng. Cândido Santos, agradeço pela disponibilidade e pelos interessantes diálogos que tivemos.

Ao grupo de amigos que criei em Aveiro e que sempre me acompanhou nesta jornada académica, os “mitras”. Uma parte importante de mim existe graças a todos eles.

Por fim e não menos importante, agradeço profundamente aos meus pais toda a força e incentivo que sempre me deram ao longo deste anos, pois sem eles não teria conseguido chegar até aqui.

palavras-chave

Registo clínico electrónico, exames complementares de diagnóstico, pEHR, aplicações portáteis.

resumo

A presente dissertação analisa os conceitos associados a um registo pessoal electrónico de informação médica, capaz de ser transportável e passível de satisfazer a necessidade de mobilidade dos cidadãos da sociedade moderna.
Pretende-se apresentar um diferente processo de recolha, arquivo e acesso a informação complementar de diagnóstico, passando de um arquivo disperso e muitas vezes inacessível para um acesso electrónico integrado “de bolso”.
Desta forma, é elaborado um estudo sobre dispositivos móveis de arquivo capazes de suportar o registo, assim como analisadas tecnologias de desenvolvimento de aplicações portáteis.
São abordadas técnicas de aquisição e armazenamento persistente de informação de índole clínica, centrada principalmente em exames complementares de diagnóstico.
Após um processo de estudo e selecção de ferramentas com vista à implementação das funcionalidades pretendidas, é proposta uma arquitectura de registo médico pessoal transportável e desenvolvida uma aplicação demonstrativa do conceito.

keywords

Electronic medical records, complementary diagnostic tests, pEHR, portable applications

abstract

This dissertation examines the concepts associated with an electronic personal medical information record that can be portable and capable of meeting the mobility needs of the modern society citizens.
It presents a different process of collecting, archiving and accessing additional diagnostic information, changing from a scattered and often inaccessible archive to an integrated electronic access “in your pocket".
Thus, is produced a study on mobile storage devices capable of supporting this kind of record, as well as an analysis on portable applications developing technologies.
Are also presented techniques for acquisition and persistent storage of clinical nature information, mainly focused on complementary diagnostic tests.
After a study and selection process of tools to implement the desired functionalities, we propose an architecture for a portable personal healthcare record and develop a demo application of the concept.

Índice

Índice de Tabelas

Índice de Figuras

Capítulo 1 - Introdução

1.1 Enquadramento

Os cuidados de saúde e o seu respectivo sistema de prestação, são necessidades fundamentais da sociedade e, estão fortemente embutidos nas sociedades modernas. Muitos intervenientes, organizações e indivíduos, fazem parte do sistema nacional de prestação de cuidados de saúde, desde os prestadores de serviços, passando pelos sistemas financiadores (seguros e subsistemas), e pelas organizações que monitorizam o estado da saúde e tentam melhorá-lo. Os pacientes procuram sempre um adequado tratamento das suas necessidades de saúde e querem que as suas informações pessoais sejam preservadas de modo seguro e protegido. [1]

Um dos maiores desafios que os pacientes enfrentam no sistema de saúde actual é a lacuna que existe no acesso integrado aos seus registos de informação médica. A qualidade da prestação de cuidados de saúde beneficiaria com a capacidade de agregar um de conjunto informação médica dos pacientes, via arquivo centralizado ou integração no momento do acesso, de forma a estar disponível para os pacientes e os prestadores de cuidados de saúde devidamente habilitados para o efeito.

Ao olharmos para a actual realidade de prestação de cuidados de saúde verificamos que uma parte muito significativa da informação médica produzida para um paciente resulta dos seus exames complementares de diagnóstico. Um exame complementar de diagnóstico é um sistema de apoio à decisão médica, executado com o intuito de ajudar o diagnóstico de uma determinada patologia. São exemplos deste tipo de exames, radiografias (produção

imagiológica), análises clínicas (exames laboratoriais), electrocardiogramas (sinais eléctricos), etc.

Estes exames são actualmente suportados por dois formatos, analógico e digital. Se o formato analógico, papel e película de filme, era predominante até há pouco tempo, hoje em dia o formato digital está em franca expansão. Exemplos desta realidade são as modalidades imagiológicas e os electrocardiogramas (ECG), assim como alguns relatórios médicos que já circulam segundo standards digitais [2] [3] [4].

Com o surgimento e proliferação dos registos médicos electrónicos torna-se também possível fornecer aos pacientes um acesso e controlo sobre a sua informação clínica. [5] Comparando com os registos médicos em formato analógico, os dispositivos digitais começam por facilitar o transporte e armazenamento dos dados graças à redução da sua dimensão física. Contudo, a realidade mostra-nos que o paciente continua a ser o principal actor no processo de arquivo e transporte da informação, tornando-se preponderante no processo de partilha de dados entre profissionais de saúde, sejam eles exames complementares de diagnóstico ou relatórios de consultas de especialidade.

Nos últimos anos tem havido grande actividade à volta dos registos clínicos electrónicos e da sua consequente adopção. No entanto, os registos médicos pessoais não tiveram o mesmo grau de atenção (e discussão), pelo menos na realidade Portuguesa. E enquanto um registo electrónico de saúde centralizado funciona como um serviço virado essencialmente para os profissionais de saúde, um registo de saúde pessoal recebe, organiza e arquiva informação médica acessível ao cidadão. Este controla o acesso aos dados e assume a responsabilidade de facultar a informação a profissionais de saúde responsáveis pela prestação de cuidados de saúde à pessoa.

Actualmente muita da biografia biomédica ainda não descreve com detalhe as potencialidades e mais valias de um sistema electrónico do tipo "ePHR- electronic Personal Health Care Record". Muitas vezes referenciado na literatura apenas como registo médico pessoal e não como registo médico portátil. Neste estudo objectivamos desenvolver conhecimento na área do registo médico pessoal e transportável.

Os registos de saúde pessoais são muito mais do que meros arquivos distribuídos, estáticos e desconexos contendo informação do paciente; eles combinam conhecimento, dados e ferramentas para que os pacientes se tornem um elemento activo na gestão da

informação relativa às suas condições de saúde. É de todo o interesse do paciente fazer com que este repositório de informação se torne, integrador, dinâmico e pró-activo, potenciando maiores benefícios no processo de prestação de serviços clínicos.

Os exames complementares realizados em centros de diagnóstico especializados são normalmente entregues como cópia única, não ficando muitas vezes nenhum outro registo nessas instituições (pelo menos por um período de tempo alongado), o que comportaria a necessidade de repetição do exame, em caso de perda da única cópia.

Na sequência do cenário descrito, visionamos a oportunidade de desenvolvimento de um conceito de uma plataforma pessoal digital transportável de informação médica. Comodamente usamos o seu termo inglês "pEHR – portable Electronic Healthcare Record", sigla para "Registo Médico Electrónico Pessoal e Transportável". Tratando-se de um novo paradigma de recolha, arquivo e acesso à informação clínica centrado no paciente, inúmeras questões devem ser alvo de análise, debate e especificação como, por exemplo: Em que consiste? Quem são os beneficiários? Quem o usa? Que interacção lhes proporciona? Que relação estabelecerá com os actuais sistemas electrónicos da saúde? Qual a arquitectura do sistema? Que estratégias a tomar para que se ultrapassem as barreiras de adopção do sistema?

1.2 Motivações e Objectivos

Viver em sociedade implica a todo o momento estar consciente das orientações imperantes, sendo que a evolução é o principal factor a ter em atenção. Daí a importância deste trabalho, fornecer uma proposta e tentar demonstrar a sua exequibilidade para que essa evolução aconteça.

A elaboração desta dissertação está toda ela centrada na introdução de um conceito que se pensa ser viável, útil e capaz de melhorar a qualidade da prestação de cuidados médicos e consequentemente da qualidade de vida do cidadão. Ao vislumbrar essa possibilidade, a sua apreciação e tentativa de 'prova de conceito' são, sem dúvida, desafios aliciantes que geram motivo de grande interesse, daí que se tornem as principais motivações para a realização deste estudo.

Esta dissertação apresenta como objectivos centrais o estudo e enquadramento com os conceitos a si associados. Designadamente, tecnologias de virtualização e técnicas para a criação de uma aplicação com potencialidades portáveis que seja capaz de armazenar e apresentar informação médica pessoal relativa, sobretudo, a exames complementares de diagnóstico. Um sistema capaz de funcionar a partir de um dispositivo de armazenamento transportável sem nenhum tipo de dependência da máquina hospedeira. Garantindo também que o protótipo do sistema referido satisfaz todos os aspectos relativos à protecção de dados clínicos confidenciais.

Sintetizando, ambiciona-se provar que o conceito de plataforma de armazenamento de informação clínica é viável e que é capaz de beneficiar directamente o próprio cidadão, o funcionamento da sociedade e dos serviços que prestam cuidados de saúde.

1.3 Estrutura da Dissertação

A estrutura desta dissertação foi delineada de forma a que se possa gradualmente compreender os conceitos nela envolvidos e se depreenda clara e objectivamente o trabalho desenvolvido. Começaremos por abordar os principais temas e conceitos, passando pela experimentação de tecnologia e apresentação do protótipo da aplicação. Finalmente, é apresentada uma discussão de resultados, conclusão e especulação sobre o trabalho futuro.

O capítulo 2 introduz o estado da arte tanto a nível clínico como a nível tecnológico. O propósito deste capítulo é proporcionar conhecimento sobre os principais aspectos do cenário clínico e as tecnologias utilizadas ao longo do trabalho.

Após a contextualização com os conceitos e tecnologias chave, no capítulo 3 é apresentada a proposta deste estudo, onde são abordados os seus principais requisitos e possíveis casos de utilização. É descrita ainda a arquitectura do sistema e os aspectos essenciais de implementação.

No Capítulo 4 é dada uma abordagem final ao protótipo da aplicação. É analisado o processo de escolha de ferramentas a utilizar na sua implementação e descritas as suas fases de construção. São também analisados e discutidos os resultados obtidos no seu desenvolvimento, terminando com uma demonstração das funcionalidades implementadas.

Finalmente as conclusões obtidas da implementação do sistema proposto, da discussão dos resultados e da produção desta dissertação são apresentadas no Capítulo 5, onde são também abordadas possibilidades de trabalho futuras.

Capítulo 2 - Estado da Arte

Este capítulo de estado da arte pretende introduzir e contextualizar os conceitos chave aliados aos principais cenários em que esta dissertação se focaliza. Encontra-se dividido em duas secções, a primeira relativa ao cenário clínico e a segunda dedicada ao panorama tecnológico deste trabalho.

2.1 Cenário Clínico

Nesta secção, dedicada ao cenário clínico é incluída uma abordagem aos essenciais formatos de registos de informação clínica (com relevância no pEHR), aos standards e normas de portabilidade de informação clínica digital. Finaliza-se com uma análise exemplificativa do espaço em disco ocupado por ficheiros digitais de exames complementares de diagnóstico.

2.1.1 Registos de Informação Clínica

É necessário clarificar, compreender, como é que os pEHR (*Portable Electronic Healthcare Records* - Registos de Saúde Electrónicos) podem funcionar de modo a beneficiar não só os seus portadores mas também os seus prestadores de cuidados de saúde.

Antes, é importante diferenciar este conceito de registo médico pessoal de outros tipos. Distingue-se em primeiro lugar dos registos analógicos (papel, película, etc.) por ser uma

plataforma completamente electrónica. No entanto não prejudica este formato analógico, mas antes, fornece uma forma de o complementar. Este tipo de registo (analógico) é o mais antigo, comum e largamente usado, em todas as áreas da medicina. É um método de baixo custo, viável e que na maior parte das vezes não necessita de nenhum tipo de equipamento para se tornar legível. O seu arquivo poderá ser da responsabilidade da instituição (centralizado) ou do paciente (descentralizado).

O registo digital poderá ser assente num arquivo informático, fixo e centralizado de informação. Este tipo de modelo arquiva informação normalmente relativa apenas a uma única instituição onde está presente. Além da capacidade de arquivo poderá fornecer também outras funcionalidades como a codificação, impressão, cópia de segurança, entre outras. Este tipo de arquivo está, como o analógico, sujeito a danos físicos, assim como a falhas informáticas que potenciam perdas de informação.

Temos também o modelo de informação centralizado que é editado e acedido através da internet via *web browser*. Este tipo de modelo já suporta informação de variadas fontes, incluindo informação inserida pelo indivíduo ou automaticamente por sistemas já existentes nas instituições. Apresentam a vantagem de poderem ser acedidos a partir de qualquer localização desde que esta possua um computador com ligação à internet. Dois grandes projectos recentes que implementam este tipo de modelo são o *Google Health* da Google [6] e o *Health Vault* da Microsoft [7], que em comum criam um sistema centralizado em que o utilizador controla o que é arquivado e quem pode aceder à informação. São duas soluções gratuitas, fáceis de utilizar com menus de interacção simples. A segurança é um factor crítico e estas afirmam-se ser tão seguras como serviços de *online banking*. Para fornecer e facilitar a interoperabilidade com terceiros sistemas, fazem uso de standards electrónicos, como por exempo, o CDA (Clinical Document Architecture) e o CCR (Continuity of Care Record), o que permite a integração directa das suas informações. [8]

Finalmente, ao contrário dos sistemas centralizados, temos um tipo de registo que conjuga as características dos registos pessoais analógicos e descentralizados, com a facilidade de armazenamento do registo electrónico e com a mobilidade proporcionada pelo uso de um dispositivo de armazenamento móvel. Podendo basear-se assim num suporte digital portátil, tal como um CD, DVD, disco ou drive *flash*, como é o caso desta proposta.

Existem já algumas soluções baseadas em drives *flash*, principalmente nos EUA, onde podemos armazenar informação médica, não sob a forma de arquivo de informação e historial

de exames e relatórios, mas essencialmente vocacionadas para situações de emergência [9]. Através destes dispositivos é possível consultar facilmente informações médicas relevantes, tais como, identificação, elementos de contacto, historial de doenças, alergias, etc. Estes dispositivos possibilitam uma intervenção rápida da equipa médica de emergência e, acima de tudo, mais eficiente a partir do momento que se conhecem informações cruciais sobre a pessoa que necessita dos cuidados.

2.1.2 Registo Médico Pessoal Transportável

Para o cidadão comum, o pEHR tem um conjunto de potenciais e benefícios. Um dos mais importantes é fornecer uma melhor capacidade de acesso à sua informação médica, tornando-se num veículo de melhorar a partilha desta informação. Os pacientes podem assim fornecer o acesso à informação a quem lhes presta cuidados de saúde, contribuindo para melhorar a qualidade do serviço prestado. Quanto maior a variedade de informação que é passível de ser inserida no registo, mais vantagens pode o utilizador retirar dele.

Este conceito é aplicável a qualquer indivíduo. No entanto, um paciente mais interessado no seu cuidado de saúde como, por exemplo, um paciente com algum tipo de patologia crónica, usufruirá mais do facto do seu historial clínico estar organizado e facilmente acessível no pEHR.

Os prestadores de cuidados de saúde também saem beneficiados com este registo, pois terão mais elementos para elaborar um diagnóstico e tomarem melhores decisões. Mais ainda o pEHR permite encurtar o tempo que se despende numa análise de elementos informativos muitas vezes transmitida oralmente ou de forma desorganizada pelo paciente. Desta forma, estamos a melhorar substancialmente a relação entre o profissional de saúde e o paciente, assim como a sua eficiência; promovendo uma relação de continuidade mais do que numa relação esporádica, ultrapassando a barreira limitativa da comunicação cara a cara.

A utilização de um sistema deste tipo passa por um processo de educação das pessoas, elucidá-las da importância da gestão pessoal dos seus registos médicos, para que possam perceber realmente a utilidade deste registo e dessa forma retirarem dele uma mais valia.

Estes factores complexos a nível humano e organizacionais, podem ser um factor de aceleração ou de entrave à adopção do conceito pEHR. Muitos desafios surgem, pois estamos a falar de várias organizações envolvidas que desempenham papéis fulcrais, mas acima de tudo falamos de uma realidade que é óbvia e o principal interessado e beneficiário, o cidadão.

Tal como qualquer outro sistema de informação clínica, os pEHR devem garantir a privacidade e veracidade dos dados assim como fornecer um suporte para autenticação dos usuários de forma a promover a confiança dos utilizadores destes sistemas. Actualmente existem variadíssimas formas de implementar segurança e confiança, passando pelos sistemas baseados em criptografia de chaves pública até elementos biométricos.

Garantir a veracidade dos dados incluídos no registo está dependente de sistemas de validação e por bom senso não devem ser alterados pelo utilizador. No entanto, da mesma forma que oralmente se descreve, a quem presta cuidados de saúde, os antecedentes clínicos com o detalhe que se pretende, também a informação contida no registo pode ser filtrada pelo utilizador.

No caso de um registo médico portátil é da responsabilidade e do interesse do indivíduo a actualização dos conteúdos. A sua interface deve ser simples, intuitiva e acessível ao comum cidadão que nem sempre este possui conhecimentos clínicos aprofundados ou domina a terminologia médica, factores susceptíveis de reduzir a motivação para actualizar o registo médico portátil. É de facto um desafio tornar estes dispositivos úteis para o cidadão, onde os dados devem ser inseridos e apresentados de forma a que o indivíduo compreenda como interagir, dinamizar essa informação, potenciando um benefício na prestação de cuidados de saúde.

2.1.3 Portabilidade de Informação Clínica

A tentativa de portabilizar a informação clínica levou ao desenvolvimento de muitas aplicações e standards de "transporte digital" de informação. Desta forma, para uma interoperabilidade máxima foram criados vários formatos de suporte electrónico e normas de exames complementares de diagnóstico. Estas normas definem um conjunto de regras de arquivo e comunicação deste tipo de informação digital. Permitem que os documentos

produzidos em formato digital sejam facilmente trocados entre máquinas e instituições que partilhem determinada norma. A circulação de informação entre organismos sai beneficiada com esta situação.

No caso dos Electrocardiogramas (ECG), existem alguns esforços para estabelecer standards electrónicos independentes de formatos proprietários de fabricantes. A realidade é que existe uma larga variedade de formatos o que dificulta a padronização para interoperabilidade. Os formatos mais reconhecidos e utilizados são: o aECG (*annotated* ECG), desenvolvido pela *Food and Drug Administration* (FDA) e *Health Level Seven* (HL7), e o SCP-ECG (*Standard Communications Protocol of Computerized Electrocardiography*). [3] [10].

Na área de imagiologia médica a norma *Digital Imaging Communications in Medicine* (DICOM) foi criada de forma a padronizar os formatos de arquivo e as comunicações de imagens médicas. Tal normal impõe-se como o grande standard mundial nesta área, daí que a compatibilidade de um pEHR com DICOM seja crucial. Esta norma garante a comunicação integrada, permitindo que a informação médica (imagens digitais) esteja inserida numa rede internacional de comunicação médica, ou seja, que um exame seja visualizado exactamente da mesma forma em qualquer ponto do planeta (previne diagnósticos errados ou duplos por conta de equipamentos diferentes). [2] [11]

Os relatórios do paciente sob a forma de documento electrónico encontram-se essencialmente em dois formatos distintos e muitas vezes visto como concorrentes, o *Clinical Document Architecture* (CDA) da HL7 e o *Continuity of Care Record* (CCR) da *ASTM International*. Estes formatos suportam essencialmente informação sob a forma de texto que envolve uma interpretação humana, assim como partes estruturadas para reconhecimento por software [4] [12].

2.1.4 Volume de Informação

Com o advento dos formatos digitais em informação clínica, nomeadamente exames complementares de diagnóstico, existe uma inerente necessidade de armazenamento de um grande número de procedimentos realizados pelo cidadão ao longo da sua vida. O volume de informação (i.e. o tamanho dos ficheiros produzidos) varia bastante, dependendo de inúmeros

factores. Desta forma, torna-se complexo estimar o seu valor pelo que nos cingiremos a abordar alguns casos críticos, nomeadamente o volume de informação dos procedimentos imagiológicos.

Em imagiologia médica, os volumes de dados produzidos variam bastante de acordo com o tipo de procedimento e a modalidade. Para além disso, aspectos como, o tipo de compressão e formato de encapsulamento utilizado são decisivos. Com base em repositórios de imagens e em estudos efectuados é possível observar exemplos do espaço ocupado por alguns tipos de exames.

Por exemplo, um exame eco-cardiográfico pode ocupar algumas centenas de *Megabytes* (MB) da mesma forma que um exame de tomografia computorizada com vários cortes, que produz muitas imagens, pode ultrapassar um *Gigabyte* (GB) de espaço. [13]

No entanto, o tamanho ocupado pode ser reduzido significativamente, através de algoritmos reversíveis de compressão de imagem. Mais ainda, existem algoritmos irreversíveis, onde a taxa de redução de tamanho é superior a dez vezes, não comprometendo significativamente a qualidade de diagnóstico da imagem. Quanto maior a taxa de compressão, menos espaço de armazenamento será necessário. [13] [14].

Através de alguns exemplos, que usam algoritmos de compressão de imagem (*jpeg2000*), é possível observar os diferentes tamanhos ocupados por alguns tipos de estudos: [15]

- Uma ressonância magnética (MRI) cardíaca : 41 MB
- Uma tomografia computacional (CT) de uma aorta dilatada : 167 MB
- Um CT abdominal normal nas fases arterial e venosa : 102 MB
- Um estudo normal CT dinâmico de alta resolução 4D (3650 imagens) : 898 MB
- Um CT de uma fractura pélvica : 57 MB

A necessidade de espaço de armazenamento está ainda dependente de outras variáveis, por exemplo, o tipo de paciente, as suas patologias, a regularidade com que realiza exames e os que pretende arquivar. São factores que diversificam, daí que o espaço de arquivo necessário tenha de ser adequado a diferentes perfis.

2.2 Cenário Tecnológico

Na sequência da contextualização introduzida no cenário clínico (secção 2.1) e de forma a complementar essa informação é necessário introduzir alguns conceitos tecnológicos relevantes.

No desenvolvimento de uma plataforma móvel de aquisição de informação, dois aspectos importantes para esta temática serão abordados: a análise de dispositivos móveis de armazenamento digital, de forma a proporcionar um suporte para o arquivo de ficheiros e, o desenvolvimento de um dispositivo que permita a aquisição, arquivo e visualização dessa informação. Com esse intuito, e sem perder de vista o conceito de portabilidade, a contextualização com tecnologias de virtualização e técnicas para criação de aplicações portáteis serão abordadas nesta secção.

2.2.1 Dispositivos Móveis de Armazenamento

Pela secção 2.1.4 verifica-se a necessidade de espaço de armazenamento para os ficheiros clínicos. Tendo em conta a necessidade de mobilidade, é necessário considerar alguns dispositivos de armazenamento móvel. Desta forma, nesta secção são analisados e comparados os principais dispositivos desta categoria.

Desde os anos 80 e da disquete 5”1/4, os dispositivos de arquivo portáteis têm vindo a proliferar, evoluindo para o CD, *Zip-Drive*, DVD, e para recentes tecnologias, as unidades de disco *flash* e os discos rígidos portáteis.

Actualmente, os dispositivos de armazenamento móvel são pequenos e robustos. Proporcionam a capacidade de conter e transportar dados facilmente, quer num ambiente de trabalho, quer em mobilidade.

Grande parte das pessoas não arrisca guardar dados importantes exclusivamente no disco rígido do seu computador, recorrendo a dispositivos portáteis para que possam armazenar uma cópia de segurança do seu material, fácil de transportar e de transferir ficheiros.

Embora com muitas vantagens associadas, estes dispositivos também têm as suas limitações. São mais susceptíveis a perca ou roubo, o que os torna mais inseguros para transportar dados pessoais e sensíveis como, por exemplo, os dados clínicos de um paciente. Nesta medida, cuidados complementares de segurança devem ser considerados.

A capacidade de mover dados em qualquer lugar e a qualquer altura, sobrepõe-se muitas das vezes a essas limitações. Até mesmo ambientes baseados em infra-estruturas complexas de armazenamento podem tirar benefícios deste tipo de tecnologia. À medida que as suas capacidades aumentam e o seu preço desce, este tipo de armazenamento torna-se cada vez mais interessante para empresas e para o utilizador comum. Alguns exemplos de dispositivos de armazenamento móvel serão seguidamente analisados.

2.2.1.1 CD/DVD – Blu-Ray – HD DVD

Sendo suportes baratos e com elevada disponibilidade, os CD (*Compact Disc*) e DVD (*Digital Versatile Disc*) são uma referência em estratégias de armazenamento ou cópias de segurança. São suportes que são usados facilmente para criar e distribuir apresentações, instruções ou conteúdos multimédia para diversos fins. As unidades de gravação são também de baixo custo, sendo as da tecnologia DVD retro compatíveis com o formato CD.

As tecnologias agora em emergência, *Blu-ray* e HD(*High Defenition*) DVD, permitem até 8.5 GB de espaço de armazenamento, mas são ainda incompatíveis e disputam um lugar de supremacia no mercado actual. Também por este motivo os leitores e gravadores destes formatos ainda não se tornaram um standard, dividindo as marcas produtoras de equipamentos e levando a que se pratiquem preços ainda bastante elevados.

A dimensão e aparência física destes dispositivos são bastante semelhantes e praticamente imperceptível. Na Figura 1 é possível observar a aparência física de um CD.

Figura 1 – Aparência física de um CD.

2.2.1.2 Discos Rígidos (Não Portáteis e Portáteis)

O disco rígido externo é actualmente um dos acessórios informáticos de armazenamento portátil mais utilizado. Há dois tipos de discos externos desta natureza, os de 3,5" e os de 2,5", equivalentes aos tamanhos daqueles que podemos encontrar internamente num *desktop* (computador de secretária) ou num computador portátil. Na realidade são internamente como eles, dispositivos de armazenamento não volátil guardado em superfícies magnéticas rotativas. A diferença reside em se encontrarem num invólucro que os reveste. Este invólucro permite protege-los e contém a interface necessária para que sejam ligados ao computador, através de uma ficha USB (*Universal Serial Bus*) ou *Firewire*.

Estes dispositivos fornecem um elevado espaço para armazenamento, com o custo dependente do seu tamanho físico, sendo o de 2,5" pelas suas mais reduzidas dimensões o mais caro. Porém, devido ao seu baixo consumo energético não necessita de alimentação externa, esta é fornecida pela porta onde está ligado para transferência de informação. Por outro lado as unidades de 3,5" obrigam a que uma alimentação externa ao computador tenha de ser utilizada, essa dependência torna-a não tão portátil face à de menores dimensões.

Na Figura 2, é possível observar a aparência de um disco externo de 2,5".

Figura 2 - Disco Externo 2,5" [16]

2.2.1.3 Dispositivos *Flash* USB

O único requisito de hardware necessário no computador para que um dispositivo *flash* USB possa funcionar é uma porta USB, sendo estes dispositivos automaticamente detectados pela maior parte dos computadores. A porta USB versão 2.0 é bastante comum actualmente em qualquer computador, além de ser retrocompatível com antigas normas do formato.

A memória *flash* é uma memória não volátil, um tipo específico de memórias EEPROM (*Electrically Erasable Programmable Read-Only Memory*). Hoje em dia é usada em variadas aplicações, desde câmaras digitais passando pelos leitores de áudio digitais até aos telemóveis e computadores portáteis.

A sua utilização mais comum é sob a forma de *Pen Drive* (Figura 3) que conjuga as reduzidas dimensões da memória *flash* com a compatibilidade praticamente universal da porta USB, criando um dispositivo de armazenamento simples, cómodo e bastante útil.

A portabilidade, as capacidades de simples manuseamento e a inúmera variedade de formas são as suas maiores vantagens. Actualmente 64GB de armazenamento já se encontram disponíveis no mercado e esta capacidade está em constante progresso, desta forma fornece um excelente suporte para um arquivo de dados.

Não está sujeita aos danos de sujidade que podemos encontrar num DVD nem a danos por efeito de campos magnéticos que afectam os discos rígidos (HDD). Como não possuem as partes móveis são muito mais resistentes ao impacto, pelo seu próprio design, que as torna robustas. Obviamente o invólucro em que se encontram define muita dessa robustez, mas até mesmo o comum, em plástico, as torna bastante sólidas e confiáveis. É também de salientar o baixo consumo energético de uma unidade deste género ligada na porta USB.

Podem vir também dotadas de outras funcionalidades para além do armazenamento de informação, incluindo o suporte à codificação interna (por *hardware*) do seu conteúdo e até mesmo identificação por leitor biométrico de impressão digital.

Figura 3 – "Pen-drive" USB

2.2.1.4 Estudo Comparativo e Conclusões

Nesta secção pretende-se analisar comparativamente os suportes de informação referidos anteriormente. Considerando como principais factores de comparação: a dimensão física, peso, capacidade de suporte, velocidade de leitura/escrita e custo, a Tabela 1 apresenta essas informações para cada um dos dispositivos.

	DVD	HD-DVD Blu-Ray	Disco Rígido Portátil USB (2.5")	Pen-Drive USB
Dimensão média	12cm (raio) x 1.2mm	12cm (raio) x 1.2mm	13 x 2 x 9 cm	7 x 2 x 1 cm
Peso médio	80 g	90 g	250 g	10 g
Capacidade máxima	8,54 Gb (dual)	50/30 Gb (dual)	750 Gb	64 Gb
Velocidade de Leitura	27 MB/s (20x)	36 MB/s (8x)	60MB/s	60MB/s
Velocidade de Escrita	27 MB/s (20x)	36 MB/s (8x)	60MB/s	60MB/s
Custo médio	0,21 €/Gb	0,6 €/Gb	0,25 €/Gb	1,5€/Gb

Tabela 1 – Comparativo entre suportes físicos de informação digital

Tendo em conta a análise e comparativo anteriores entre os diversos suportes físicos de informação digital (Tabela 1), considera-se que o mais adequado para o transporte de conteúdos clínicos pessoais seja a drive *flash* USB "pen-drive". Tendo em conta a sua relação entre tamanho físico e quantidade de dados capaz de armazenar, o seu custo é reduzido, tornando-se apropriada para o tipo de utilização pretendido. Ao mesmo tempo que é um suporte em constante expansão de capacidade, não existe um limite para este, surgindo a todo o momento no mercado novas capacidades a melhores preços.

2.2.1.5 Tecnologias Emergentes

O futuro do armazenamento móvel tende para um novo formato, já actualmente disponível, as unidades de estado sólido (*SSD – Solid State drive*) [17] e a tecnologia USB versão 3.0, ainda em desenvolvimento, que trará melhorias significativas principalmente ao nível da velocidade. [18]

Os discos rígidos da tecnologia SSD que recorrem à tecnologia *flash*, semelhante à usada nas drives *flash* USB, são mais vantajosos por inúmeras razões, sendo as mais importantes, o seu tempo de acesso reduzido, a eliminação das partes móveis dos convencionais discos de tecnologia magnética, um menor peso, maior tolerância a altas temperaturas, reduzido consumo energético e velocidade de transferência muito superior (100 MB/s na gravação e 80 MB/s na leitura) [19]. A tecnologia está em forte expansão e desenvolvimento, actualmente novas *drives* estão a ser apresentadas com uma elevada frequência, existindo o potencial para que os tradicionais discos rígidos sejam substituídos por unidades desta tecnologia. A curto prazo é complicado afirmar isto, provavelmente as duas tecnologias vão partilhar simultaneamente o mercado por um longo tempo. Devido ao seu actual custo, apenas encontramos uma grande aplicação em computadores portáteis de pequeno tamanho, onde os aspectos dimensão e consumo são importantes.

Quanto ao USB versão 3.0, revela uma importante mudança face à tecnologia usada presentemente, i.e. a USB 2.0. À semelhança do ocorrido na transição do USB 1.1 para o USB 2.0, existirá um progresso significativo da velocidade de transferência, cerca de seis vezes mais, passando dos actuais 60 MB/s para 400 MB/s [18]. Este incremento justifica-se pelo uso de pares de linhas dedicados independentemente para download e upload, permitindo transferências simultâneas. Tal como aconteceu na mudança anterior, a versão 3 continuará retro-compatível com as anteriores normas do formato.

No futuro, considerando que o suporte escolhido como o mais adequado para o pEHR tire partido desta tecnologia (USB 3.0) então, este terá um enorme benefício acrescentado devido ao aumento das velocidades de transferência. Este aumento, promoverá menores tempos no arquivo e acesso à informação influenciando directamente a usabilidade do dispositivo, principalmente quando se arquivam/acedem grandes volumes de informação.

2.2.2 Virtualização

A virtualização tem ganho muita atenção em todas as organizações que procuram responder a novos desafios, melhorando a eficiência das suas operações e aumentando a sua capacidade de resposta a novas condições. É essencialmente o isolamento de recursos computacionais, separando-os em diferentes camadas e ambientes de execução, permitindo uma maior flexibilidade na sua gestão. [20] [21]

Existem diferentes tipos/níveis de virtualização, os mais relevantes para a contextualização deste estudo serão seguidamente introduzidos.

2.2.2.1 Virtualização de Hardware

O primeiro conceito de virtualização surge normalmente associado ao *hardware*. Através de programas específicos, simulando componentes físicos de um computador, podemos gerar máquinas virtuais (*Virtual Machines*), possibilitando assim que vários sistemas operativos sejam instalados em cada uma delas e usados simultaneamente.

As vantagens prendem-se obviamente com o facto de serem eliminadas as questões de incompatibilidade entre aplicações e sistemas operativos, pois através da máquina virtual podemos ter outro sistema operativo disponível e executar normalmente alguma aplicação que não tenha sido desenvolvida para o sistema operativo base da máquina. E mesmo instalando máquinas virtuais que corram o mesmo sistema operativo que a máquina base, podemos facilmente usá-las como plataformas para um ambiente de teste, eliminando riscos em testar novas aplicações ou actualizações. Estas vantagens são mais direccionadas a utilizadores finais de computadores, mas também em servidores podemos encontrar grandes vantagens. Desde logo, evita-se a necessidade de subservidores que não utilizam todos os recursos computacionais da máquina onde se encontram, substituindo-os por uma distribuição de processos num menor número de computadores, aproximando a capacidade da máquina aos seus níveis de aproveitamento, evitando-se o subaproveitamento de hardware. A redução destes requisitos também diminui os custos de manutenção, o espaço físico e os consumos energéticos.

Algumas soluções comerciais existem mas, uma das mais utilizadas é o software de virtualização da VMware Inc., que dispõe de inúmeras soluções direccionadas para estações de trabalho e servidores, onde as licenças de *workstation* rondam os 200 euros. [22]

Este software, à semelhança de outras soluções similares, permite criar uma camada de software directamente no sistema operativo da máquina, contendo uma ferramenta de monitorização que aloca os recursos de *hardware* de forma dinâmica e transparente. Deve-se chamar à atenção que não se trata de um sistema emulador, pois temos interacções a um nível mais baixo, onde o processador chega por vezes a executar directamente o código da máquina virtual e, apenas quando isto não é possível, o código é convertido.

Permite a coexistência simultânea de sistemas operativos e aplicações num único computador, onde cada um tem acesso aos recursos de que necessita apenas quando deles precisa.

2.2.2.2 Virtualização da Apresentação

Na virtualização da apresentação, como o próprio nome indica, a aplicação na verdade corre noutra máquina remota e o que vemos e temos acesso no nosso computador é o ecrã da máquina onde essa aplicação está a correr. Algo como um acesso remoto a uma aplicação centralizada, de modo a permitir que vários utilizadores usufruam simultaneamente do mesmo sistema e sem que interfiram entre si.

Esta tecnologia é considerada de apresentação, pois só a camada de apresentação da aplicação, i.e. as suas telas e caixas de diálogo, é que são transmitidas ao utilizador. Quanto ao lado do utilizador, são capturadas as entradas do rato e teclado e devolvidas à máquina que hospeda a aplicação.

2.2.2.3 Virtualização de Aplicações

As aplicações dependem do sistema operativo para uma variedade de serviços, sejam eles drivers, alocações de memória ou registo de bibliotecas. Tratar problemas de compatibilidade entre as aplicações e o sistema operativo é simples usando os tipos de virtualização já referidos. Mas se o problema se focar em duas aplicações distintas, como por exemplo um conflito entre bibliotecas dinâmicas (DLL) ou outras incompatibilidades, o problema é resolvido virtualizando a aplicação num outro dispositivo ou máquina que possui instalada a sua cópia de operação. Os utilizadores podem assim aceder às potencialidades da aplicação, sem necessitarem que esta esteja instalada na máquina física. Mais ainda, as aplicações não necessitam de direitos de administração do sistema operativo base para instalação de eventuais componentes. Assim, o programa pode ser executado normalmente e as características específicas de cada aplicação são interpretadas directamente pela máquina do utilizador através de uma camada virtual criada em paralelo. Permitindo inclusive, que funcionem simultaneamente várias aplicações sem que se afectem entre si (Figura 4).

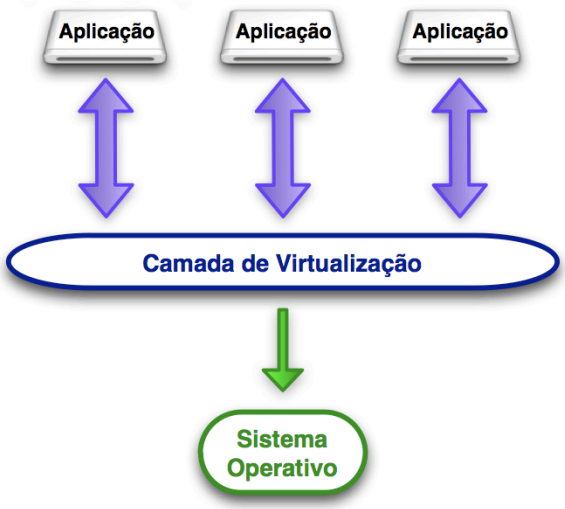

Figura 4 - Virtualização de aplicações

Em vez de virtualizar um completo sistema operativo virtualiza-se apenas uma aplicação, através da criação de um ambiente virtual em que a aplicação trabalha e reage como se estivesse a interagir com o sistema operativo e os seus recursos. Na realidade, quaisquer que

sejam as alterações efectuadas, nunca afectam o sistema operativo onde a aplicação está a ser executada, já que a aplicação está a interagir com uma camada de virtualização especialmente criada para o efeito.

Uma aplicação após ser instalada gera uma série de dependências, as mais comuns são:

-Configurações e outras informações no registo do sistema (considerando o sistema operativo *Windows*).

-Faz uso de variáveis do sistema (*environment variables*).

-Pode instalar certos componentes de maneira especial, como por exemplo registar determinado binário como serviço do *Windows*.

-Pode modificar ficheiros existentes quando é instalada e pode aceder e modificar os mesmos durante sucessivas execuções.

-Finalmente uma aplicação pode também depender de sistemas de base de dados, o que pode implicar a necessidade de instalar, configurar e popular a mesma. [23].

A camada de virtualização tem assim a capacidade de interceptar, interpretar e redireccionar estas operações para uma localização virtual, não fazendo nenhuma alteração ao sistema operativo onde foi lançada a aplicação. Ao mesmo tempo, informa a aplicação e eventualmente o seu instalador que todas as alterações foram realizadas com sucesso, para que estes possam continuar a sua normal operação. Todos os pedidos da aplicação e respectivas respostas são agora da responsabilidade da camada de virtualização. Se o recurso necessário está disponível no contentor virtual então, é fornecido à aplicação. Se não, estes pedidos são redireccionados para o sistema operativo base.

Tal como na virtualização de hardware, inúmeras vantagens advêm deste tipo de virtualização. Além de uma melhoria da segurança, há também uma utilização mais eficiente do hardware, com uma escalabilidade e fiabilidade melhorada. Mais ainda, facilita futuras actualizações e migrações, e os custos de equipamento e licenças fica reduzido.

Exemplos de soluções disponíveis para implementar este tipo de virtualização passam pelo *Citrix XenApp* e pelo *Microsoft Aplication Virtualization*, também conhecido como *Microsoft SoftGrid*. [24] [25]

2.2.3 Aplicações Portáteis

As aplicações portáteis (*Portable Aplicattions*) inserem-se no âmbito da virtualização de aplicações, mas neste caso específico a aplicação é executada directamente a partir de um suporte externo, um CD-ROM, disco externo ou *flash* drives USB. OS CD-ROM e outros dispositivos de leitura apenas, estão limitados por não ser possível utilizar aplicações que neles necessitem de guardar informações. Daí que, quando se fale em guardar informação no suporte se assuma que este permite essa funcionalidade.

Hoje em dia, o termo *"portable application"* refere-se na maior parte dos casos a uma aplicação que corre a partir de um dispositivo portátil, ex. *flash drive* (a vulgar "*Pen* USB*"*) ou disco externo. Permitindo às pessoas terem acesso às suas aplicações mais usadas em qualquer computador, sem que nele se encontrem realmente instaladas.

Alguns exemplos de aplicações portáteis largamente utilizadas, são ferramentas de trabalho e produtividade como o "*OpenOffice Portable*", os *browsers* como o "*Mozilla Firefox Portable*", clientes de email como o "*Mozilla Thunderbird*" ou os clientes de texto e voz como o "*Skype Portable*". Existem ainda soluções que integram uma série de aplicações num mesmo contentor virtual, desde leitores de multimédia a jogos, com todos os dados pessoais e de configuração disponíveis, permitindo utilizar qualquer computador como sendo o pessoal sem fazer nenhuma alteração ao mesmo. Um exemplo deste tipo de solução é a plataforma desenvolvida pelos colaboradores da *portableapps.com suite*. [26]

Actualmente já existe um conjunto muito diversificado de aplicações disponíveis a qualquer utilizador, precisando apenas de um computador e de um dispositivo portátil. Porém, requerem normalmente o uso de um sistema operativo específico (Windows, Mac OS, Linux, etc.).

Se a aplicação não foi projectada para funcionar dessa forma (portátil), é necessário um processo de redesenho da aplicação e novos testes terão de ser executados. De modo a evitar esta necessidade, é possível utilizar um processo de virtualização e criar um "invólucro virtual" que force a aplicação a seguir determinadas especificações. É importante assim que este invólucro inclua rotinas que controlem o lançamento da aplicação e, durante a sua execução, permitam interceptar todos os pedidos de operações ao sistema operativo. Pedidos que

tentem modificar o estado, de ficheiros ou registos, da máquina onde o dispositivo está inserido são redireccionados para o dispositivo portátil.

Aproveita-se para fazer um importante reparo sobre a expressão "*portable*" ou portátil. Muitas vezes fala-se, em tecnologias de informação, no conceito de código portátil que é distinto do conceito que utilizamos neste estudo para nos referirmos a aplicação portátil. O primeiro incide sobre a capacidade de poder reutilizar o código de uma aplicação por forma a permitir corre-la em diferentes plataformas. Por outro lado, o segundo refere-se à capacidade de executar uma aplicação sem dependências de outros softwares, ou máquinas virtuais como é o caso deste estudo em particular.

2.2.3.1 Criação de uma Aplicação Portátil

A criação de uma aplicação portátil, segue normalmente alguns passos que passo a descrever. Existem suites de software capazes de virtualizar uma aplicação existente de modo a torna-la portátil, uma destas soluções é a *ThinApp* da VMware Inc. [27]. As licenças para este produto rondam os 4000 euros, incluído uma licença do software de virtualização *"workstation"*, já referido, e 50 licenças de cliente e suporte. [28]

Esta solução de *software* permite "portabilizar", i.e. tornar portáteis, a generalidade das aplicações. No entanto, há casos em que isso não é possível como, por exemplo, quando uma aplicação necessita de um driver específico de um dispositivo, quando possui uma protecção contra cópia baseada em hardware ou exige software que venha incluído em sistemas operativos que não se encontre num pacote separado.

O que esta aplicação fundamentalmente faz é capturar as alterações que a instalação e utilização da aplicação provocam no sistema operativo, algo semelhante a um "antes e depois". Depois, através da informação recolhida, cria algo semelhante a um contentor executável que vai conter a aplicação e toda a informação que precisa de ser redireccionada, incluindo ficheiros do utilizador para que sejam guardados localmente como, por exemplo, na drive USB onde a aplicação vai correr, ao invés do disco rígido da máquina. No final do processo um executável é criado e a aplicação torna-se assim portátil (Figura 5).

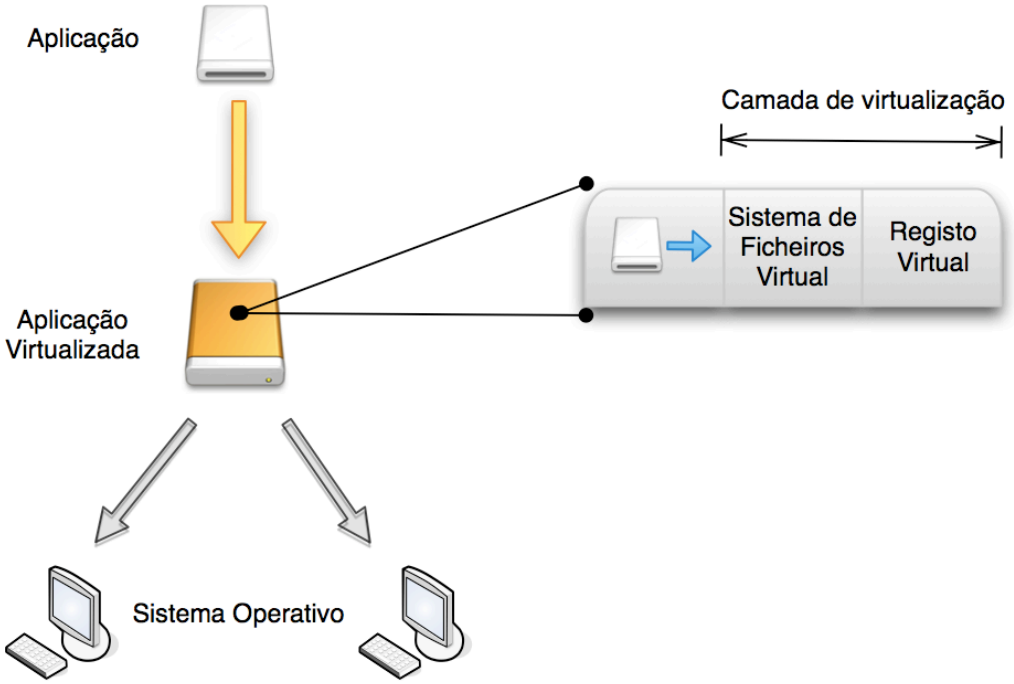

Figura 5 – A aplicação é virtualizada junto com todas as suas definições num só executável isolado do sistema operativo.

É necessário ter algum cuidado quando submetemos software proprietário ou que necessite de uma licença específica a este processo. A não ser que a licença do software o permita, a distribuição deste tipo de aplicação após o processo de virtualização pode ser considerada ilegal. Aconselha-se portanto o uso de aplicações de software aberto, cuja alteração de código e redistribuição é livre.

2.2.4 Ferramentas de Desenvolvimento de Aplicações Portáteis

Outra solução para obter uma aplicação com características portáteis, similares às da virtualização é, desenvolver de raiz um dispositivo que evite dependências de terceiros softwares, que necessitem de procedimentos de instalação. Na realidade trata-se do conceito de aplicação "*standalone*" , algo que não é de todo um conceito recente, mas que tem andado um pouco "esquecido" na última década face às características das novas plataformas de desenvolvimento e respectivas dependências.

Quando falamos da possibilidade de executar uma aplicação dependendo apenas da presença do computador, estamos indirectamente a referir a presença de um sistema operativo nessa máquina. Existem hoje em dia inúmeros sistemas operativos, mas segundo dados estatísticos, relativos à cota de mercado, a percentagem do *Windows* é claramente superior à dos outros sistemas, com podemos constatar no gráfico da Figura 6.

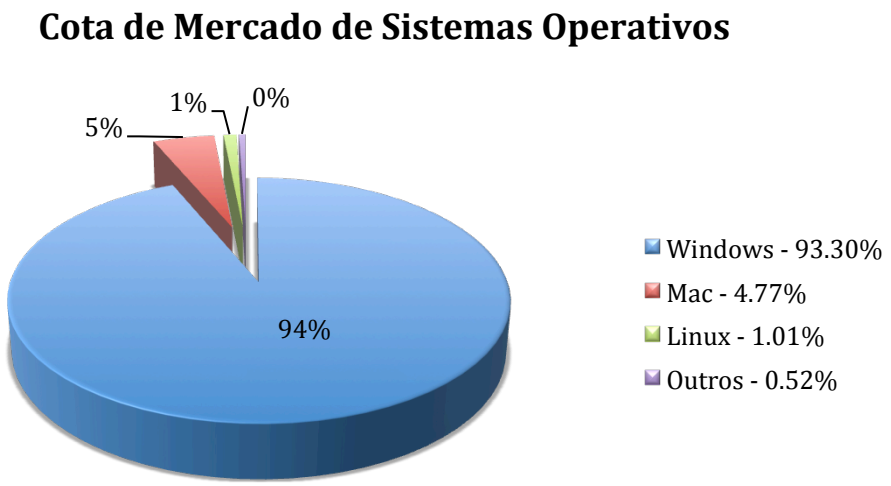

Figura 6 - Gráfico gerado para a cota de mercado de diferentes sistemas operativos em computadores pessoais em Agosto de 2008 [29]

Sendo que idealmente o desenvolvimento da aplicação deveria ser orientado a todos os sistemas operativos para total interoperabilidade e independência. Numa primeira fase, e baseado na observação da cota de mercado decidimos efectuar estudo do estado da arte de ferramentas de desenvolvimento orientadas ao sistema operativo *Windows*.

Começando por avaliar as principais linguagens de programação, vamos analisar quais as mais indicadas para os requisitos definidos e quais as interfaces de desenvolvimento mais adequadas para desenvolver uma aplicação com as características pretendidas.

Uma das linguagens mais populares do momento é a Java. [30] Esta linguagem tem a especial particularidade de ter sido desenvolvida com o intuito de ser portável entre sistemas operativos. Para isso recorre a uma máquina virtual, a "JVM – *Java Virtual Machine*", que necessita de estar instalada no sistema operativo em que a aplicação irá ser executada. Esta

característica oferece à linguagem Java um interessante poder de desenvolvimento multi-plataforma. Mas ao mesmo tempo também determina uma dependência obrigatória, a necessidade de uma máquina virtual Java no sistema. O utilizador final de uma aplicação desenvolvida em Java precisa de instalar o "*JRE - Java Runtime Environment* – Ambiente de execução Java" que lhe permite correr a aplicação. Para além de que, conforme a versão de desenvolvimento usada na aplicação, poderá ter de ser instalado um *JRE* mais actual para que esta funcione correctamente. [30] [31]

Centrando a discussão no desenvolvimento de soluções de *software* em ambiente Windows, as ferramentas mais actuais estão incluídas na plataforma .NET (*dot Net*). Esta engloba um conjunto de linguagens e interfaces que permitem desenvolver aplicações bastante versáteis. À semelhança da linguagem Java, esta plataforma obriga a presença de uma dependência, o .NET *Framework*. Acrescentando o facto de as várias versões não serem compatíveis, o que obriga a instalar a versão correcta do *Framework* associada à aplicação desenvolvida. [32]

Na Tabela 2 podemos observar um comparativo entre a percentagem de utilização sistemas operativos Windows no mercado e a sua respectiva inclusão e compatibilidade com as versões do *.NET Framework*.

Versão do Sistema Operativo	% de PCs	Versão .NET				
		1.0	1.1	2.0	3.0	3.5
Windows 95	0.00%	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Windows NT	0.38%	Download	Download	Não disponível	Não disponível	Não disponível
Windows 98	0.30%	Download	Download	Download	Não disponível	Não disponível
Windows Me	0.18%	Download	Download	Download	Não disponível	Não disponível
Windows 2000	1.66%	Download	Download	Download	Não disponível	Não disponível
Windows XP	73.55%	Download	Download	Download	Download	Download
Windows Vista	23.82%	Compatível	Compatível	Incluída	Incluída	Download
Windows 7 Beta 1	0.1%	Compatível	Compatível	Incluída	Incluída	Incluída
PCs com a versão .NET incluída		23.93%	23.93%	23.93%	23.93%	0.11%
PCs compatíveis com a versão .NET		100%	100%	99.62%	97.47%	97.47%

Tabela 2 - Tabela gerada a partir da informação de Dezembro de 2008 ajustada apenas para incluir os computadores Windows [29]

Conclui-se que, as possibilidade de encontrar um computador equipado com determinada versão do *.NET Framework* são relativamente baixas. A maior parte dos computadores pré-*Windows Vista* em locais públicos (bibliotecas, cafés, hotéis) assim como os dos hospitais ou clínicas provavelmente não terão todas as versões do *Framework* instaladas. Obviamente que haverá excepções caso uma outra aplicação instalada no computador requeira a presença de determinada versão para funcionar. Mas, no geral, não poderemos considerar aplicações desenvolvidas com base em algum *.NET Framework* realmente portáteis, pois estarão sempre dependentes de determinada versão pré-instalada no computador para poderem funcionar correctamente.

As plataformas de desenvolvimento anteriores funcionam em moldes semelhantes, obrigando o utilizador à instalação do seu software de suporte respectivo. No entanto, existe uma plataforma de desenvolvimento alternativa ao "mundo Microsoft" que não obriga a utilizar uma máquina virtual nem à presença de um Framework. Trata-se do Delphi, uma ferramenta de desenvolvimento da *Borland*. O seu suporte a vários sistemas operativos (Windows, Linux) torna-a numa ferramenta eficaz para o desenvolvimento de aplicações portáveis a vários níveis, abstraindo-se do sistema operativo e de dependências de outros softwares.

Mas e as outras ferramentas acima referidas estarão impedidas de produzir uma aplicação portátil? Na verdade, originalmente não. No caso da linguagem Java, a máquina virtual, torna-se um entrave à possibilidade das aplicações desenvolvidas correrem no sistema operativo sem a sua presença. Porém, já existem esforços no sentido de também portabilizar a própria plataforma que suporta a máquina virtual Java [33]. No entanto, tratam-se de projectos paralelos que não possuem nenhum tipo de suporte oficial da *Sun*, a empresa responsável pela linguagem. A actual instabilidade destas soluções exploratórias, leva-nos a concluir que ainda é difícil ultrapassar a dependência da máquina virtual, o que impossibilita um desenvolvimento estável de aplicações portáteis com esta linguagem.

Quanto às aplicações desenvolvidas na plataforma *.NET*, a dependência do seu *Framework* é incontornável. Aquando do aparecimento da plataforma, o ambiente de desenvolvimento integrado (IDE) lançado para a suportar foi o *Microsoft Visual Studio .NET,* que surgiu no seguimento do *Microsoft Visual Studio 6*. Esta última plataforma não tem nenhuma relação de dependência do *.NET Framework* e uma escolha cuidada dos

componentes gráficos permite desenvolver aplicações *standalone*, daí que também se tenha considerado como possível ferramenta de desenvolvimento de uma aplicação pEHR.

2.2.5 Futuro da Virtualização e das Aplicações Portáteis

O futuro da virtualização passará pela computação em nuvem "*Cloud Computing*", referente a um estilo de computação em que os recursos são fornecidos de forma virtual pela internet como se de um serviço se tratasse, em que o utilizador não tem de se preocupar com a manutenção do serviço. Tecnologia esta que está a ser fortemente adoptada e que corrobora a evolução e sucesso da virtualização.

No caso das aplicações portáteis, as vantagens são inúmeras como já foi referido, havendo desde já uma grande quantidade de aplicações a serem desenvolvidas na sua versão normal e também na versão portátil, evitando que seja necessário um desenvolvimento complementar, muitas das vezes, assegurado por entidades diferentes do produto original. No futuro esta política deverá ser largamente adoptada por equipas de desenvolvimento de *software*. Oferecendo, ainda mais, a possibilidade aos cidadãos de terem acesso a todos os seus documentos e suas aplicações mais utilizadas em qualquer computador, mesmo que não possua ligação à internet.

2.2.6 Segurança de Dados Portáteis

As informações de saúde do paciente são dados muito delicados. Os registos clínicos contêm informações sobre o historial de saúde do cidadão e devem ser confidenciais, deste modo, a segurança da informação é um factor crítico pois, caso seja comprometida põe em risco a privacidade pessoal. A crescente mobilidade dos cidadãos e da portablidade digital das informações médicas faz com que novas estratégias de segurança do seu armazenamento tenham de ser exploradas.

Desde que começaram a surgir os primeiros dados informáticos, que certamente se pensou em como poder arquivá-los e protegê-los. Abordaremos agora de forma sucinta algumas soluções para protecção de informação digital.

O primeiro conceito que nos surge neste contexto é obviamente o de cifragem de dados. Garantindo que estes não se corrompem, da mesma forma que se controla o seu acesso e a sua privacidade. A cifragem do disco é uma tecnologia que nos permite codificar dados guardados, num qualquer suporte, protegendo a informação. Este processo é normalmente realizado com o auxílio de um software de cifragem.

Um exemplo de um software com estas características é o *TrueCrypt*, bastante utilizado hoje em dia e interessante pelo facto de ser de código aberto e gratuito [34]. Enquadrando esta aplicação na temática das aplicações portáteis verificamos que este possui uma limitação relevante. Muito embora o modo *"portable"* exista no *TrueCrypt* este necessita que se possuam privilégios de administrador na máquina onde o utilizamos para que os drivers de cifragem possam ser utilizados. Num cenário em que ciframos informações num suporte amovível para uso em máquinas pessoais não causa grande condicionamento. No entanto, num cenário em que pretendemos circular a informação por várias máquinas, onde podemos não ter esses privilégios, torna-se um problema aceder à informação [35].

Para além da cifragem de dados sobre a forma de volume, outros métodos são comummente utilizados. A incorporação de um conjunto de ficheiros num só arquivo de forma comprimida é um desses métodos.

Actualmente, um reconhecido formato de compressão de dados é o arquivo Zip, que pode ser usado como caso de estudo para arquivo e segurança de dados portáteis.

Um arquivo zip contém um ou mais ficheiros que poderão estar organizados, ou não, por directorias. Usando algoritmos de compressão estes podem encontrar-se comprimidos, reduzindo o seu tamanho, ou simplesmente armazenados sem nenhum tipo de compressão. Existem imensos softwares no mercado que suportam o formato zip, suporte esse, que já se encontra também embutido na maior parte dos sistemas operativos actuais.

Os recursos do arquivo zip permitem-lhe além da compressão a codificação dos ficheiros que armazena. A última versão do standard zip já suporta algoritmos de codificação recentes, como o AES-256, tornando-se mais confiável. As especificações actuais do formato

(Zip64) incluem também suporte a ficheiros grandes (maiores que 4GB) e a novos métodos de compressão. [36]

Como vantagem, este tipo de armazenamento para além de proteger a informação também possibilita a sua compressão, diminuindo o tamanho total ocupado pelo arquivo. É desvantajoso por ser possível ver o seu conteúdo (nomes de ficheiros, pastas, tamanhos, etc.), o que não oculta totalmente a informação. Porém não é possível extrair os dados codificados do seu interior sem a respectiva password de codificação.

Capítulo 3 - Proposta

Neste capítulo apresenta-se a proposta de arquitectura do protótipo de sistema que intitulamos de pEHR – "*Portable Electronic Healthcare Record*".

Primeiramente serão expostos alguns problemas do cenário clínico já introduzido nos capítulos anteriores, que motivaram a realização deste estudo, assim como as propostas para a resolução dos mesmos. Haverá também uma referência com maior detalhe aos nossos objectivos centrais, incluindo uma descrição dos cenários possíveis de utilização e, dos seus intervenientes.

3.1 Contextualização

Começaremos por uma abordagem de contextualização a um conjunto de aspectos problemáticos que poderão ser facilmente identificados quando abordamos a actual situação da criação de informação médica em formato digital ou analógico, nomeadamente, dos exames complementares de diagnóstico.

O prestador de cuidados de saúde, recorre com frequência a ferramentas auxiliares de apoio ao diagnóstico, gerando muita informação que necessita de ser arquivada num suporte específico e analisada no momento da sua criação ou posteriormente. Desde logo, o primeiro aspecto, que nos parece óbvio, está associado à dispersão das fontes produtoras dessa informação, pois a maior parte das pessoas recebe serviços de muitos prestadores de cuidados de saúde, das mais variadas índoles e especialidades.

Esta proliferação de informação por diversas fontes leva a que se acumulem exames e consequentes relatórios nos mais diversos suportes e locais, muitas das vezes sem ligação entre si. Sendo um exemplo muito simples da situação, o caso do paciente que recorre a um determinado especialista para diagnosticar uma patologia e decide consultar uma segunda opinião em outro local. Ambos vão produzir informação clínica relevante para o seu diagnóstico e não possuem uma interligação que lhes permita consultar o que outro fez e que conclusões tirou. Mais ainda, este processo de prestação de cuidados de saúde é também partilhado por diferentes especialidades cooperantes.

Na sequência do exemplo anterior, verificamos que, poderá existir um meio de comunicação entre os especialistas, quanto mais não seja o próprio paciente. Este funciona muitas vezes como meio de transporte, arquivo e troca de informação partilhada. Tendo em sua posse uma cópia física dos eventuais exames e relatórios, está em condições de facultar, ou não, a sua consulta a terceiros.

O paciente é o responsável por guardar pessoalmente toda a sua informação num local próprio de sua escolha e gerir a sua organização. A falta da disponibilidade para o fazer de modo adequado pode levar a que facilmente esta informação se torne desorganizada para o próprio.

Outro problema que o cidadão pode encarar é o desaparecimento ou perca dessa informação. Esta fica assim inacessível para suporte futuro à prática clínica, já que a cópia era única e há necessidade de requerer uma nova cópia do exame, gastando assim recursos numa duplicação que poderia ser evitada.

Os exames que se encontram arquivados centralmente na instituição poderão ser consultados novamente sem uma necessidade de duplicação, mas isto debate-se com outra limitação associada à falta de mecanismos ágeis de comunicação (protocolada) entre as instituições. Assim, na prática só conseguimos ter acesso em tempo útil a um exame no interior da instituição onde este foi realizado. Sendo muitas vezes totalmente impossível garantir um acesso inter-institucional, obrigando a uma nova instância de exame. Este sistema de fragmentos de informação dispersos e desconexos é um importante entrave à melhoria da prestação de cuidados de saúde. Numa situação extrema como, por exemplo, uma situação de catástrofe, os serviços públicos ficam dependentes de um outro meio confiável de acesso à informação médica.

Voltando a centrar a temática no paciente como meio de transporte de informação entre instituições, esta questão é também ela problemática. O transporte dos exames poderá ser um problema, quer pela sua quantidade como pela sua transportabilidade, dadas as diferentes dimensões físicas que adquirem. Num panorama regional em que o tempo de deslocação e a distância são curtas pode não parecer um grande incómodo. Mas tendo em conta a sociedade de constante mobilidade em que hoje nos encontramos, com constantes viagens e deslocações de curto e longo curso, o acesso à informação médica fica muito limitado.

Passando dos cenários Paciente/Instituição para o plano concreto dos exames surge outra questão pertinente, a da degradação do suporte físico da própria informação. No formato analógico (papel, película) verifica-se facilmente a degradação do material com o tempo, que leva a que a informação fique total ou parcialmente ilegível. No entanto, o formato digital, embora mais imune, também poderá ter situações de degradação como, por exemplo, com o tempo e a utilização os dispositivos de armazenamento (cd/dvd) podem sofrer danos físicos que proporcionem erros ou até mesmo a completa impossibilidade de leitura do suporte.

3.2 Apresentação

Identificadas e descritas algumas das realidades problemáticas, tentaremos provar que é possível ultrapassá-las. Para isso, planejamos o desenvolvimento de um conceito de plataforma pessoal digital transportável de informação médica orientada a exames complementares de diagnóstico.

Idealmente, o sistema deve conter o máximo de informação possível, quer relativamente ao indivíduo em si como paciente, quer toda a informação médica adquirida por várias fontes ao longo do tempo. Quanto mais compreensível e organizada estiver a informação no sistema, mais útil se torna essa informação para o paciente e para o prestador de cuidados de saúde. Embora não haja uma lista do que deve ser incluído num registo deste tipo, verificamos que a maior parte da informação clínica gerada provém dos exames complementares de diagnóstico, daí que se considerem um ponto essencial a incluir na proposta de pEHR.

A informação é um importante elo de ligação entre todos os intervenientes na prestação de cuidados de saúde, incluindo o próprio paciente, para que todos possam interagir com sucesso. A mobilidade crescente dos cidadãos e a responsabilidade de todos os que fornecem tratamentos médicos, tem levado a um sistema globalizado de informação médica. Face às restrições dos sistemas centralizados, os sistemas médicos de informação apoiados em arquitecturas distribuídas começam a fazer todo o sentido, com vantagens a todos os níveis, quer financeiro (infra-estruturas de arquivo e comunicação), quer a nível técnico (conectividade e interoperabilidade).

Centrando todo o modelo de gestão de conteúdo no paciente e permitindo a troca de informação segura entre este e os profissionais de saúde, permitimos que as pessoas tenham um papel mais activo na prestação dos seus cuidados de saúde. Pretende-se assim um meio que, através de uma interface de simples utilização permita uma recolha e um armazenamento seguro de informação. Objectiva-se um repositório para informação médica pessoal, mas também um complemento para que os profissionais possam gerir melhor as condições de avaliação e tratamento.

3.3 Definição da Proposta e Requisitos

Da análise da secção 2.2.1 relativa aos dispositivos móveis de armazenamento concluiu-se que tendo em conta as suas características de dimensão e resistência física, a *"pen-drive",* é o suporte físico de eleição para o pEHR. Esta permitirá o armazenamento da informação e do software que, através da sua interface, permitirá a interacção com o primeiro.

Conseguimos assim resolver o problema da transportabilidade da informação, obtendo um suporte que é extremamente portátil graças às suas reduzidas dimensões. Mais ainda, através do invólucro de plástico oferece já um nível de resistência física considerável, que também se impõe como um critério importante.

A peça central deste meio de transporte e partilha de informação é a aplicação responsável pela interacção com os utilizadores e por todo o tratamento dos dados, também ela presente no suporte digital.

A plataforma a desenvolver tem de permitir um acesso a informação médica complementar de diagnóstico de uma forma cómoda, rápida, segura e organizada. Permitindo assim aceder aos exames a qualquer hora e em qualquer lugar. Não poderá estar dependente de nenhum outro software, necessitando apenas de um computador para poder funcionar, sendo assim uma plataforma completamente autónoma e funcional.

A já referida característica de portabilidade do software é o ponto fundamental. Para que este não dependa de nenhum outro, de modo a que seja apenas necessário introduzir o dispositivo numa porta USB e virtualmente poder funcionar em qualquer computador Windows. Pelo estudo apresentado na secção 2.2.4 sobre a cota de mercado dos sistemas operativos, optámos por centrar o trabalho apenas neste sistema operativo.

É também importante permitir que a informação apenas seja ou inserida ou acedida através deste mesmo software e não de outra forma, através da devida autenticação do utilizador. Garantindo a segurança quer dos ficheiros relativos ao exame, quer da sua respectiva informação.

A interface gráfica da aplicação deve permitir inserir e consultar a informação de forma simples e intuitiva. A organização é crucial para que rapidamente se encontre a informação que se procura.

Para auxílio em situações de emergência médica, deverá existir uma área de Emergência para que, sem requerer nenhum tipo de autenticação, se possam aceder aos dados vitais sobre o portador do sistema.

3.4 Casos de Utilização

Alguns agentes e cenários tomam parte activa da utilização da plataforma, nas figuras e desenvolvimento seguintes tentamos demonstrar quais eles são e qual o seu papel perante o sistema.

Sendo este um modelo centrado no paciente, podemos verificar, pela Figura 7 que o paciente e o seu pEHR são realmente o centro de toda a interacção, como seguidamente se descreve.

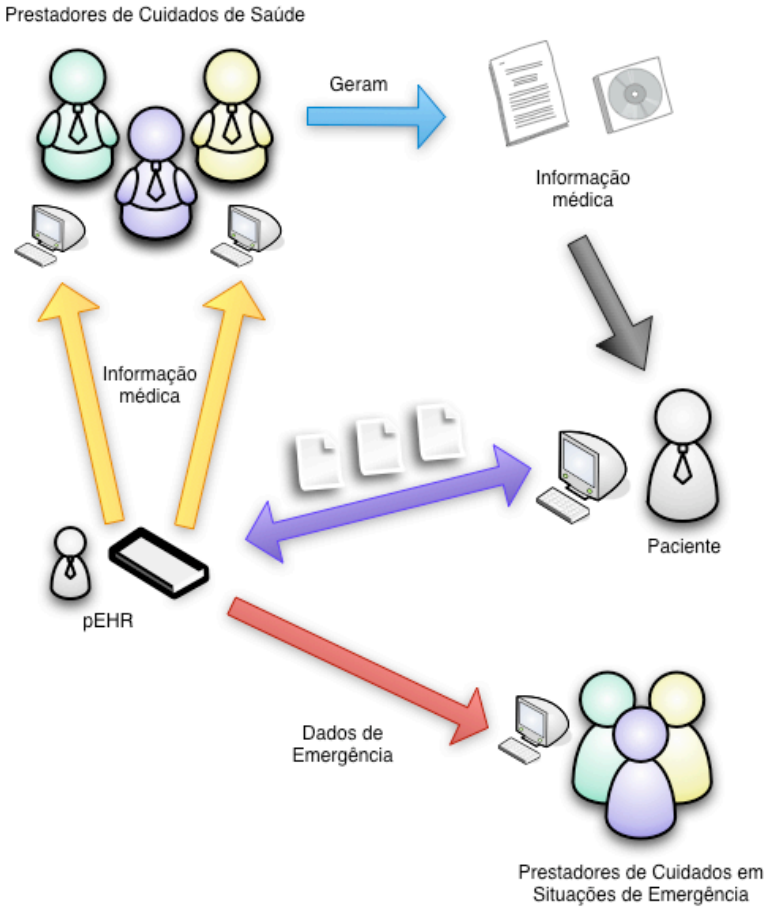

Figura 7 – Esquema representativo dos agentes e casos de utilização do pEHR.

O cidadão é o responsável pela recepção da informação médica fornecida pelos profissionais e instituições prestadores de cuidados de saúde, sejam eles hospitais, clínicas, centros de imagiologia, laboratórios ou outros locais onde tenha assumido a sua condição de paciente. Após essa recepção, encontrando-se em formato analógico terá de previamente digitalizar a informação para a arquivar convenientemente, realiza a sua introdução no sistema e insere os dados respectivos.

Na perspectiva dos profissionais de saúde eles poderão, através do pEHR, não só aceder à informação que produziram assim como à de outros profissionais de forma muito mais oportuna e organizada.

Os diferentes agentes do pEHR estão sujeitos a diferentes tipos de autenticação e consequentes permissões no sistema. O utilizador principal, o Paciente, terá acesso a toda a informação quer para visualização como para inserção, edição ou eliminação da informação. O

prestador de cuidados de saúde pode consultar a informação médica de emergência, no caso em que o paciente não está em condições de a fornecer, sem necessitar de nenhum tipo de autenticação. Além disso poderá também consultar a informação médica reservada, caso o paciente introduza a devida chave de consulta, que lhe permite apenas visualizar o conteúdo sem qualquer capacidade de alteração de conteúdos. Este processo encontra-se esquematizado na Figura 8.

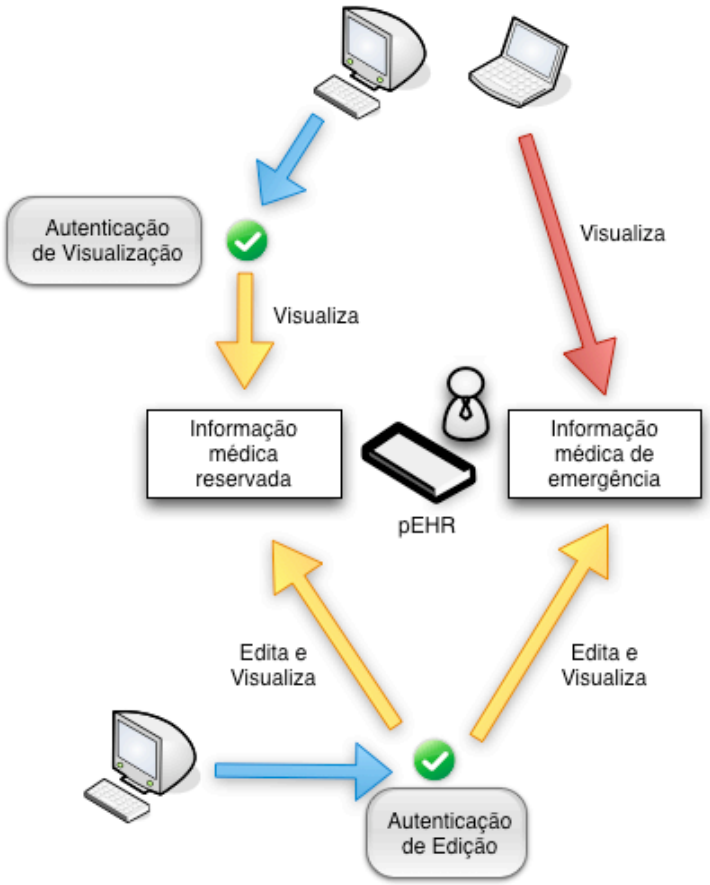

Figura 8 - Esquema relativo às diferentes autenticações e permissões do pEHR

Capítulo 4 - Desenvolvimento e Resultados

Este capítulo aborda os diferentes aspectos relativos ao desenvolvimento e implementação do protótipo demonstrador do pEHR.

Inicialmente será analisado e discutido o processo experimental desenvolvido na selecção de ferramentas que satisfaçam os requisitos descritos pelo modelo anteriormente proposto. Seguidamente, após deliberar quais as opções tomadas e com base nas ferramentas escolhidas, será apresentada uma proposta de arquitectura do dispositivo pEHR. Concluímos com uma demonstração do protótipo desenvolvido, das suas funcionalidades e dos seus respectivos aspectos de implementação.

4.1 Selecção de Ferramentas

O procedimento seguido no desenvolvimento da aplicação passou por um processo inicial de selecção das ferramentas. Neste ponto serão analisadas as várias ferramentas estudadas e os factores a ter em conta na sua selecção.

4.1.1 Arquivo de Ficheiros

O estudo inicial centrou-se na determinação da solução mais adequada para permitir o arquivo seguro dos ficheiros que contêm a informação clínica.

Inicialmente tentou-se explorar o conceito de virtualização já introduzido. Nomeadamente modificar uma aplicação existente, de forma a satisfazer os requisitos de portabilidade impostos. A situação ideal passaria por encontrar uma aplicação já testada, de código aberto, que fosse capaz de implementar a funcionalidade de arquivo seguro de ficheiros. De seguida, adaptá-la com vista a enquadrar-se nos restantes atributos pretendidos. Após a realização de trabalho de pesquisa e teste de algumas soluções que satisfizessem o pretendido, não foi obtido qualquer sucesso. Por via das dificuldades encontradas utilizar o conceito de virtualização, para atingir o propósito de criação de uma aplicação portátil, não se revelou a abordagem mais indicada para este estudo.

Decidindo centrar o estudo em outro tipo de soluções e mantendo por base de desenvolvimento a modificação de uma aplicação testada, pretende-se encontrar uma que preenche-se simultaneamente os requisitos de portabilidade e a capacidade de arquivo seguro de ficheiros. Após o trabalho de prospecção e experimentação, surge uma aplicação que se evidenciou pelas suas características, o *FreeOTFE Explorer*. [37]

O OTFE é acrónimo para "*On-the-fly encription*" – codificação criptográfica em tempo real, e trata-se de um programa gratuito, de código fonte livre e aberto. Esta solução permite o arquivo seguro de ficheiros através da criação de "volumes virtuais" cifrados. Tratam-se de unidades que funcionam exactamente como um disco normal mas onde a informação escrita é codificada, de forma segura e transparente.

Algumas das características mais relevantes deste software são: A elevada portabilidade, onde o modo de trabalho *"portable"* permite a sua utilização em diferentes computadores sem necessidade de instalação. As suas bibliotecas criptográficas diferem de outras soluções anteriormente referidas (*Truecrypt* – secção 2.2.6) pois não necessitam de privilégios de administração do sistema operativo e, desta forma, os "volumes virtuais" podem ser acedidos em computadores onde esses direitos não estão disponíveis. As suas características de segurança incluem o suporte para os principais e mais recentes algoritmos de *hashing* (SHA-512) e de codificação criptográfica (AES-256, *Twofish*), proporcionando um elevado grau de confiança.

O *FreeOTFE Explorer* é constituído por um *front-end*, implementado na linguagem derivada do Pascal (Object-Pascal) e pelos drivers de codificação escritos na linguagem C. O uso da solução *Delphi*, como introduzido na secção 2.2.4, garante a sua portabilidade.

Do ponto de vista do utilizador, a sua interface é de fácil usabilidade e acessibilidade, onde o seu assistente permite criar e aceder de forma intuitiva aos discos virtuais. É possível observar o aspecto gráfico da aplicação na Figura 9.

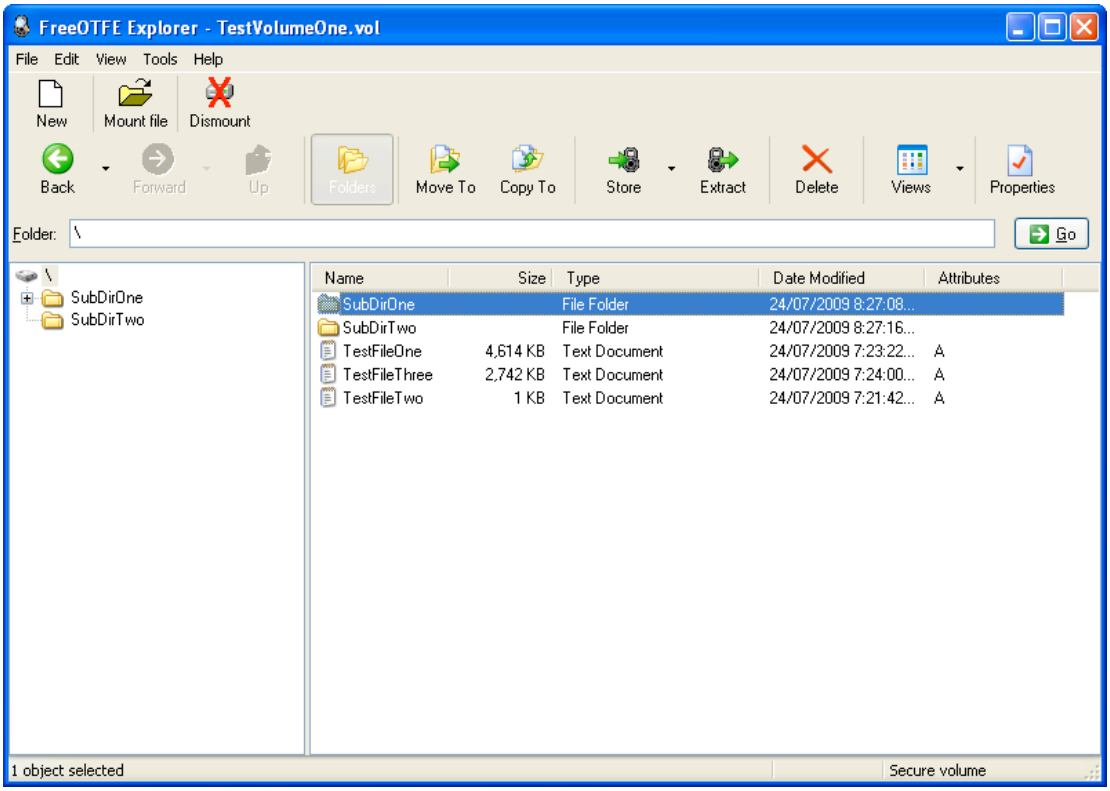

Figura 9 - Aspecto do *FreeOTFE Explorer* onde podemos observar as suas principais funções

O *FreeOTFE Explorer* revelou-se uma boa base aplicacional para desenvolver o nosso conceito de aplicação. Satisfaz os requisitos de portabilidade, garante o arquivo seguro dos ficheiros e possui código aberto.

No entanto no desenrolar dos testes de experimentação desta solução, surgiram bastantes obstáculos. Desde logo, a aplicação carecia de suporte documental depois, a inexistência de outros projectos com a mesma base de desenvolvimento, criaram dificuldades na sua compilação e costumização. Estes factores, levaram a que se optasse por abandonar tal solução. Face a esta situação, a solução baseada na escolha de uma aplicação já existente, como apoio ao desenvolvimento, não se apresentou como uma solução produtiva. A falta de soluções enquadradas neste perfil leva à necessidade de abordar a questão segundo uma diferente perspectiva.

Com base no estudo efectuado na secção 2.2.4 verificamos que através da utilização de certas ferramentas é possível o desenvolvimento integral da aplicação, ao invés de utilizar outras aplicações como base de trabalho

Continuando a análise de arquivos seguros de ficheiros, procurámos outras formas simplificadas de criar esse mesmo arquivo. Equacionamos a possibilidade do uso de um sistema de ficheiros próprio. Porém, o grau de complexidade e recursos temporais que tal solução exigia eram incomportáveis no âmbito deste trabalho.

Na sequência da análise de sistemas de armazenamento seguro de ficheiros na secção 2.2.6 e em particular do caso de estudo apresentado para arquivos de compressão, nomeadamente o arquivo zip, decidiu-se estudar a sua viabilidade como suporte à funcionalidade de arquivo de ficheiros pretendida. A sua portabilidade e características de segurança e compressão tornaram-se factores determinantes para esse estudo.

Das ferramentas analisadas, decidiu-se optar pela *ZipArchive* [38], uma biblioteca desenvolvida na linguagem C++ que oferece funcionalidades que permitem dotar a nossa aplicação da capacidade de comprimir, descomprimir e modificar arquivos zip de forma segura. Esta fornece suporte para a cifragem dos dados e variadas opções de manipulação do conteúdo do arquivo. Usando esta biblioteca foi possível criar um arquivo zip e utilizá-lo para arquivar de forma segura a informação pretendida, ao mesmo tempo que se obtém total controlo sobre o seu conteúdo.

Face a um conjunto de experimentações com resultados positivos, optou-se por escolher a biblioteca *ZipArchive* como ferramenta de arquivo seguro de ficheiros do pEHR. A sua linguagem de desenvolvimento adequa-se às nossas competências e condições de implementação, oferecendo além dos seus ficheiros de código fonte, ficheiros de projecto pré-criados para diferentes tipos de interfaces de desenvolvimento.

A questão da interface de desenvolvimento é particularmente relevante, já que se objectiva o desenvolvimento de uma aplicação "*standalone*", ou seja, unicamente dependente dela própria e do sistema operativo. Da análise da secção 2.2.4, verificamos que para uma maior compatibilidade com as máquinas existente é necessário que se eliminem o máximo de dependências de *framework*'s de desenvolvimento. Uma forma eficaz de o garantir será, utilizar uma interface de desenvolvimento que garanta que essa dependência não exista. Devido ao facto de a biblioteca *ZipArchive* se encontrar implementada na linguagem C++, o Microsoft

Visual C versão 6.0 foi a plataforma de desenvolvimento escolhida. Esta é última versão desta interface antes de se iniciar o recurso à tecnologia .NET.

4.1.2 Repositório de Informação

Após definir que a aplicação seria desenvolvida integralmente e qual a ferramenta mais adequada para implementar um arquivo seguro de ficheiros, falta encontrar uma ferramenta que nos permita desenvolver a funcionalidade de repositório estruturado de informação. Esta estrutura de dados permitirá armazenar toda a informação relativa ao pEHR, incluindo dados clínico-administrativos do paciente e informação complementar e de indexação dos diversos exames complementares de diagnóstico.

Considerámos que o uso de uma base de dados seria a melhor abordagem para armazenar e tratar a informação. As capacidades que uma normal base de dados nos proporciona são imensas o que poderá levar a que sua complexidade aumente. Dado que o uso que se pretende é algo elementar, a simplicidade de implementação e utilização do modelo de base de dados a utilizar são os principais factores a ter em conta na sua escolha.

Após um processo de análise de motores/modelos de base de dados, as questões de portabilidade foram decisivas. Com vista a implementar uma base de dados simples, a escolha da biblioteca de software *SQLite* [39], manifestou-se como a melhor opção. Seguidamente analisaremos as principais características e funcionalidades dessa biblioteca, factores críticos na sua preferência, face a outros modelos estudados.

A *SQLite* é uma biblioteca capaz de implementar uma base de dados SQL (*Structured Query Language*) transaccional, auto-suficiente, sem servidor e sem necessidades de configuração.

A maioria dos motores de base de dados SQL são implementados como um processo de servidor separado. As aplicações que necessitem aceder à base de dados, necessitam de comunicar com o servidor usando algum tipo de processo. A *SQLite* não funciona dessa forma. Com o uso desta biblioteca podemos aceder à base de dados, lendo e gravando informação, directamente no arquivo em disco. Não existe nenhum processo de servidor intermédio.

Sendo auto-suficiente, requer suporte mínimo por parte de outras bibliotecas ou do sistema operativo. Isto torna-a perfeita para a utilização em ambientes embebidos, mesmo sem a presença da infra-estrutura de um habitual computador. Daí que se adeqúe perfeitamente ao cenário de portabilidade que propomos.

A *SQLite* está desenvolvida na linguagem C e faz um uso diminuído da biblioteca standard do C, usando pouco mais de seis funções. [40] Possibilita uma fácil compilação num compilador da linguagem C standard como o que optamos utilizar, não necessitando de nenhum especial suporte já que não possui outras ligações nem dependências.

O código fonte da *SQLite* é livre e gratuito, está disponível sob a forma de um único ficheiro de código C e o seu respectivo *header file*. Desta forma é relativamente simples integrar as suas capacidades num projecto "*portable*" de maiores dimensões. Satisfazendo, desta forma, os requisitos de simplicidade de implementação a que nos propusemos.

4.1.3 Interface Gráfica

Após determinarmos as ferramentas a utilizar para dar suporte às funcionalidades de arquivo de ficheiros e de informação, necessita-se de uma ferramenta que permita desenvolver uma interface gráfica para a nossa aplicação prova de conceito.

Uma interface gráfica é essencial para que se possa demonstrar fácil e intuitivamente as funcionalidades desenvolvidas. Necessitamos de uma interface que possua um aspecto limpo e simples. A linguagem Visual Basic, assim como a sua plataforma de desenvolvimento, o Microsoft Visual Basic 6, foram as ferramentas eleitas para a desenvolver. A escolha desta ferramenta teve em conta o seu ambiente de desenvolvimento integrado, totalmente gráfico, o que facilita a construção gráfica da interface utilizador da aplicação.

Salienta-se o uso do Microsoft Visual Basic 6, à semelhança do Microsoft Visual C 6, devido a ser a última versão disponível desta interface antes do surgimento das linguagens e interfaces baseadas na plataforma Microsoft .NET. Garantimos que desta forma se elimina a dependência dos Frameworks .NET como havia sido proposto.

O estudo da secção 2.2.4 – "Ferramentas de Desenvolvimento de Aplicações Portáteis", apresenta outras alternativas mais recentes para o desenvolvimento de aplicações *standalone*

com interface gráfica. Poderia ter sido utilizada a plataforma de desenvolvimento *Delphi*, uma escolha tecnológica mais actual e portável. Porém, a falta de experiência na sua utilização bem como da linguagem Pascal, e tendo em atenção que se tratava de uma aplicação pEHR prova de conceito, levou a que se optasse pela plataforma Microsoft Visual Basic cujo conhecimento prévio nos dava garantias de sucesso.

4.1.4 Sumário e Conclusões

A escolha de ferramentas passa por um processo de estudo, análise e selecção face aos requisitos e dificuldades encontradas. Após a sua conclusão são determinadas quais seriam as mais indicadas para o desenvolvimento das principais funcionalidades planejadas.

O armazenamento de informação será implementado através de uma base de dados simples, com a ajuda da biblioteca *SQLite*. O arquivo de ficheiros será implementado através do uso de um arquivo do tipo *zip* com base na biblioteca *ZipArchive*. As funcionalidades anteriores serão desenvolvidas com o auxílio da interface de desenvolvimento Microsoft Visual C versão 6. Finalmente, com vista a demonstrar estas funcionalidades e a interacção com o utilizador, foi criada uma interface gráfica desenvolvida em Microsoft Visual Basic versão 6.

O uso de várias linguagens e interfaces de desenvolvimento leva a que a aplicação seja implementada utilizando vários ficheiros que terão de co-existir para que esta funcione correctamente. Embora isto aconteça, a questão da portabilidade do software mantém-se pelos motivos já descritos.

De seguida será apresentada em pormenor esta a arquitectura do protótipo onde serão expostos os detalhes sobre a interligação entre estes diferentes componentes, que constituem a aplicação.

4.2 Arquitectura Funcional do Protótipo

Esta plataforma está dividida em várias peças, que permitem implementar as funcionalidades que pretendíamos. A Figura 10 tenta de um modo geral ilustrar o funcionamento global da aplicação onde se evidencia a particular interacção entre os diferentes componentes.

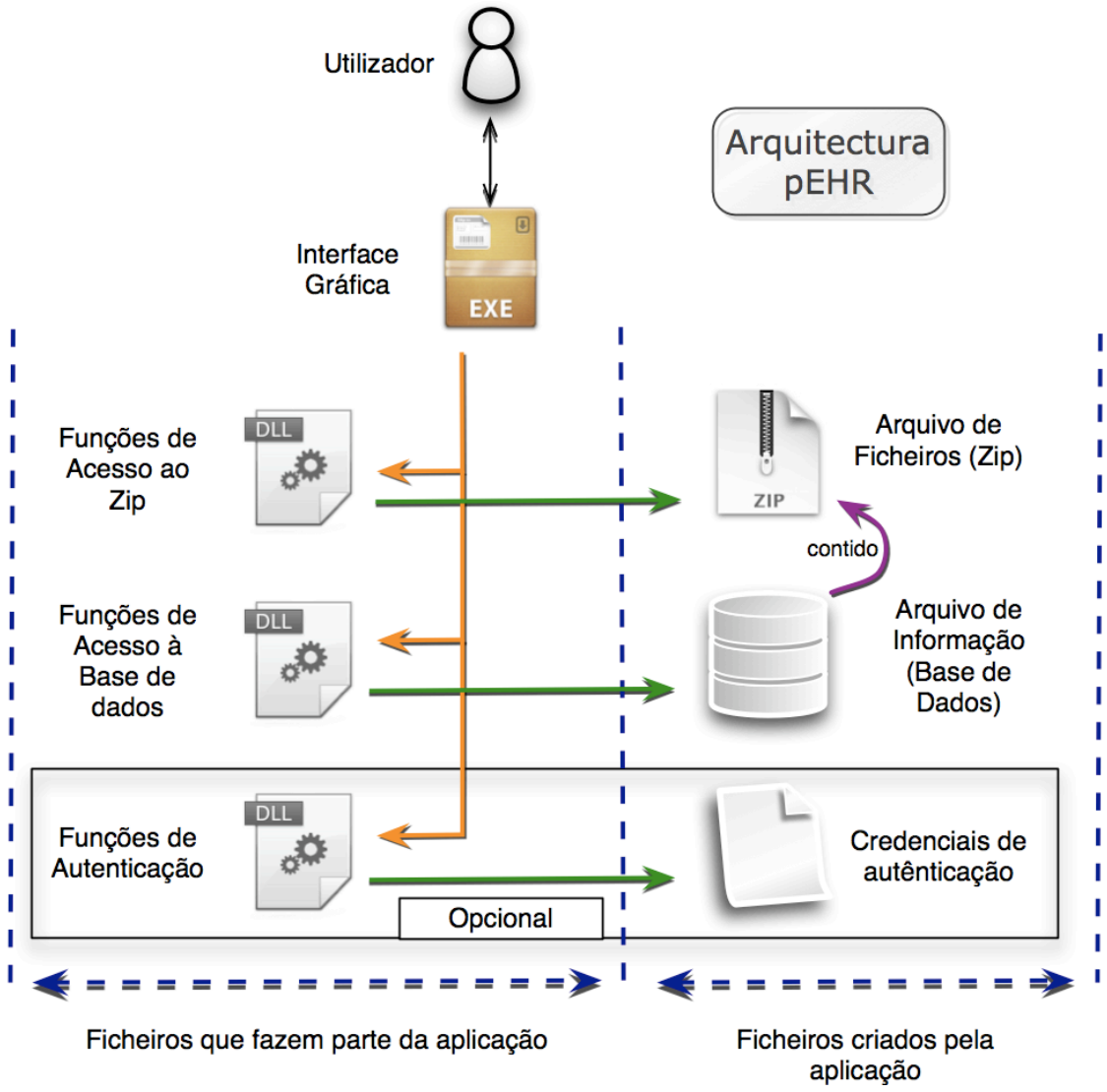

Figura 10 – Organização e interacção entre os diferentes componentes do pEHR

A interface gráfica da aplicação reside num ficheiro executável, que se limita a ser a única forma de interacção entre o utilizador e a informação que este armazena. A manipulação da informação é conseguida através do uso de bibliotecas de ligação dinâmica (DLL – *Dinamic Link Library*) onde se encontram implementadas as funções de acesso à base de dados e ao arquivo zip. Por sua vez a interface gráfica comunica com estas bibliotecas, enviando as instruções necessárias e recebendo a informação pretendida. Desta forma permite-se abstrair a interface das funções que manipulam directamente os dados.

Os vários componentes da aplicação são diferenciáveis, conforme a sua função, e podemos dividi-los em dois grupos principais.

No primeiro grupo insere-se o conjunto de ficheiros que fazem parte integrante da aplicação na sua forma inicial. Neste inclui-se o ficheiro executável (.exe) que suporta a interface gráfica do utilizador da aplicação. Assim como os ficheiros das bibliotecas (.dll) onde estão presentes as diferentes funções de manipulação dos ficheiros que a aplicação cria.

No segundo grupo temos os ficheiros que a partir da primeira utilização são criados pela aplicação. Estes são responsáveis por armazenar os ficheiros (arquivo zip) e as respectivas informações associadas (base de dados), assim como os dados pessoais e os relativos à indexação da informação clínica (base de dados).

Os ficheiros relacionados com o controlo de acessos na aplicação são de carácter opcional. Dependem da forma como esta funcionalidade está implementada, visto que, como será abordado posteriormente, duas formas diferentes de implementação são abordadas. Se esta funcionalidade se encontrar implementada recorrendo ao uso de um ficheiro externo de credenciais, é necessária a presença da sua respectiva biblioteca dinâmica (dll) responsável por criar e manipular o seu conteúdo.

A entrada e o armazenamento dos vários tipos de ficheiros e dados no pEHR é processada como esquematizado pela Figura 11.

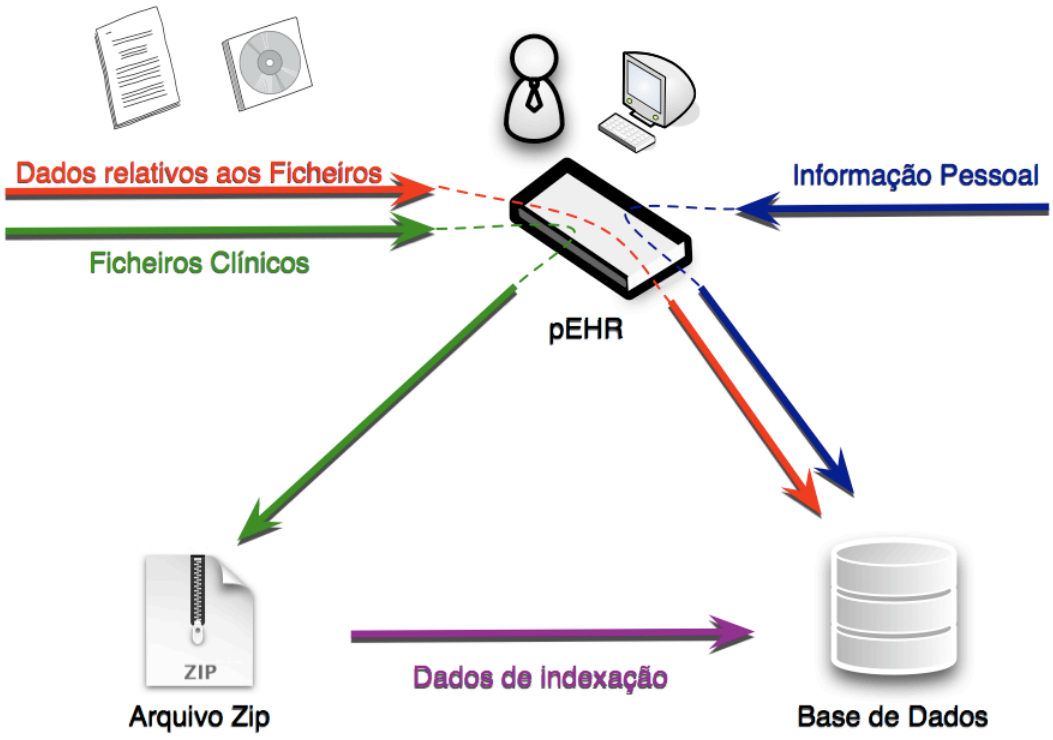

Figura 11 - Esquema relativo à organização e armazenamento dos diferentes tipos de dados

É de salientar que na base de dados, além da já referida informação dos ficheiros e da pessoal, são também guardados os dados de indexação retornados pelo Arquivo Zip. Desta forma é possível associar determinada informação ao respectivo ficheiro.

4.3 Arquitectura Aplicacional do Protótipo

No ponto anterior foi explorada a arquitectura funcional da aplicação. Neste ponto será abordado com maior pormenor a estrutura dos vários componentes, assim como as diferentes interacções realizadas entre eles.

4.3.1 Arquivo Zip e Biblioteca de Manipulação

As funções que a biblioteca *ZipArchive* nos proporciona oferecem um conjunto de operações básicas de manipulação. Através do uso de um objecto do tipo *"zip",* é estabelecida uma associação a um ficheiro zip residente em disco e sobre o qual é possível realizar operações. Foi necessário estudar/testar o funcionamento dessa ligação e determinar qual a melhor forma de a utilizar para se implementarem novas funcionalidades.

Após alterações na estrutura fornecida, foram implementadas novas funções reutilizando as existentes de forma a permitir a interacção com o arquivo zip nos moldes definidos para o pEHR. É importante salientar aquelas que são exportadas para utilização do módulo da interface gráfica, tal como a sua finalidade (Tabela 3).

Nome da Função	**Finalidade**
CriarZip	Criar o Arquivo Zip
AdFichPw	Adiciona um ficheiro ao Arquivo Zip com Password.
SubFichPw	Substitui um ficheiro no Arquivo Zip
VerificaPw	Verifica a password de um ficheiro no Arquivo Zip
ReplaceFilePw	Substitui um ficheiro no Arquivo Zip
ExtrairZipFilePw	Extrai um ficheiro com password do Arquivo Zip
DelFilePw	Apaga um ficheiro do Arquivo Zip
CheckExists	Verifica se o Arquivo Zip existe.

Tabela 3 - Funções exportadas pela Biblioteca de manipulação do Arquivo Zip

De seguida passamos a descrever sucintamente cada operação :

CriarZip – Recebe dois parâmetros, um caminho para uma directoria e um nome. Criando o ficheiro zip com esse nome nessa directoria.

AdFichPw – Recebe dois caminhos, uma para o arquivo zip, outra para um ficheiro e uma password. Adiciona o ficheiro ao arquivo protegendo-o com a password.

VerificaPw – Recebe um caminho para um arquivo zip e verifica sem alterar o seu conteúdo a password do primeiro ficheiro no interior do arquivo. Retorna um valor booleano *true* ou *false*, conforme a password seja ou não validada.

ReplaceFilePw – Recebe um caminho para um arquivo zip, o valor de um índex e um caminho para um ficheiro em disco. Substitui no arquivo zip o ficheiro localizado no índex pelo indicado pelo caminho.

ExtrairZipFilePw – Recebe um caminho para um arquivo zip, um caminho para uma directoria, o valor de um índex e uma password. Extrai, sem apagar, o ficheiro localizado no índex, do interior do arquivo zip para a directoria usando a password.

DelFilePw – Recebe um caminho para um arquivo zip, o valor de um índex e uma password. Apaga do interior do arquivo zip o ficheiro localizado no índex, usando a password.

CheckExists – Recebe um caminho para um arquivo zip, verifica a existência desse arquivo na directoria.

As funções implementadas tratam das funcionalidades básicas para manipular-mos o nosso arquivo de ficheiros. Porém para a sua melhor compreensão convém fazer uma ressalva sobre a estrutura e o tratamento específico que a biblioteca *ZipArchive* proporciona aos ficheiros arquivados no zip.

Os ficheiros dentro do arquivo poderão ser armazenados na raiz do arquivo ou organizados no interior de uma estrutura de pastas. Optou-se por utilizar o primeiro sistema, pelo que os ficheiros serão armazenados todos livremente no interior do arquivo.

Para efeitos de manipulação, com o objectivo de identificar cada um dos ficheiros, uma *key* é utilizada internamente pelo arquivo. Esta *key* é um valor inteiro único que funciona como um índex que é atribuído ao ficheiro quando este é adicionado ao arquivo. Todavia, se algum ficheiro for removido, todos os outros ficheiros de índex superior serão actualizados com um novo índex, de modo a que a sequência de valores se mantenha sempre contínua. Este índex é a principal e para a aplicação a única forma de identificar e seleccionar determinado ficheiro, daí que outras características como, por exemplo, o nome, tamanho e tipo de ficheiro são irrelevantes no processo.

Guardar o valor deste índex, bem como a sua actualização e todas as outras informações relativas ao ficheiro são da responsabilidade da base de dados. No ponto seguinte, onde a estrutura da base de dados e a sua manipulação é abordada, este tema será descrito com maior pormenor.

Outra característica que esta biblioteca possui é que, ao contrário de uma password global de acesso, permite-nos conter ficheiros protegidos por diferentes passwords dentro do mesmo arquivo, assim como ficheiros sem nenhum tipo de protecção. Na Figura 12 ilustram-se estes diferentes cenários.

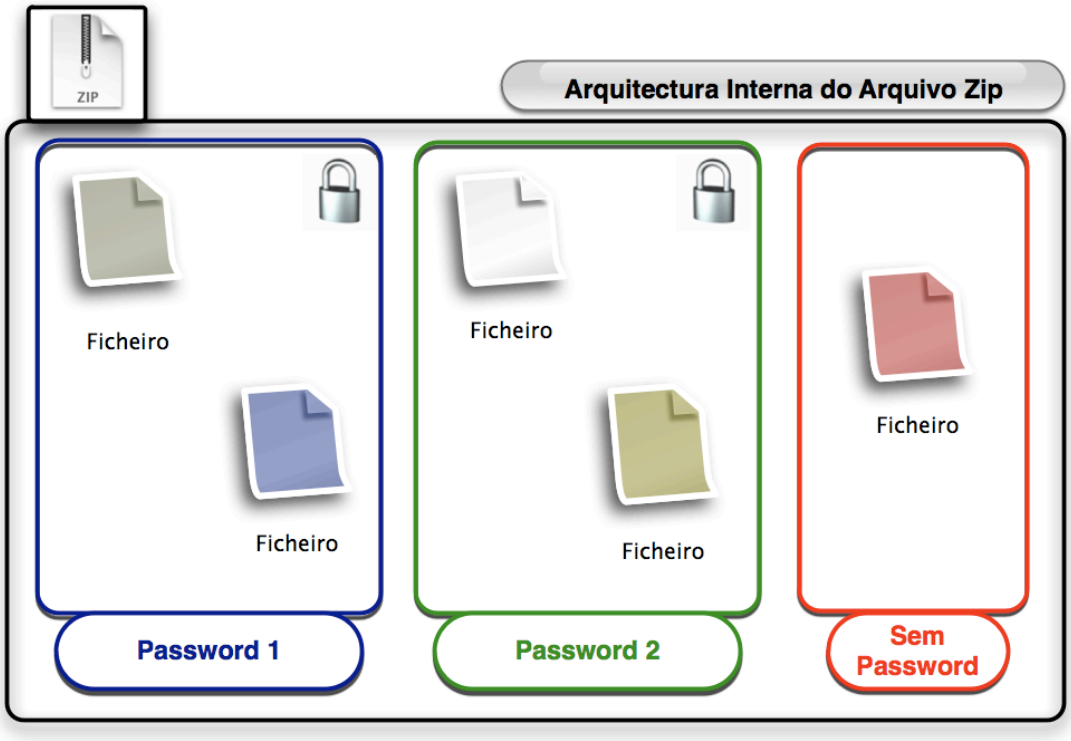

Figura 12 - Arquitectura interna de protecção por password dos ficheiros no zip

Esta arquitectura particular do arquivo permite-nos definir diferentes zonas de acesso, controladas com a possibilidade de diferentes passwords de acesso, assim como, zonas de acesso livre.

Para este estudo de pEHR apenas será utilizada uma zona protegida por password e uma zona sem password. A primeira zona irá alojar os ficheiros clínicos (exames complementares de diagnóstico) e o ficheiro de base de dados que contém as suas informações. A segunda irá alojar o ficheiro de base de dados que contém a informação clínica de emergência, para garantir que é acessível sem ser necessário uma password de acesso específica.

O método segundo a qual a password dessa zona do arquivo zip é definida, depende do método de implementação do controlo de acessos, algo que será explorado posteriormente.

4.3.2 Base de Dados e Biblioteca de Manipulação

A biblioteca implementada para manipulação da base de dados tem por base a biblioteca *SQlite*. A biblioteca *SQlite* utiliza internamente um objecto que nos permite a interacção com o ficheiro em disco que contém a base de dados. As instruções na linguagem SQL têm o nome de *statements* e foi necessário criar uma função que os recebesse directamente. A função criada recebe um *SQL statement* e o caminho para o ficheiro de base de dados. A partir daí, as funções implementadas podem reutilizar esta função para, de forma mais simples, realizarem acções na base de dados.

As funções implementadas para a manipulação da base de dados exportadas para uso através da interface gráfica da aplicação junto com a sua principal finalidade encontram-se descritas na Tabela 4.

Nome da Função	Finalidade
CriarBase	Cria um ficheiro base de dados pessoais e de ficheiros
CriarBaseEmer	Cria um ficheiro base de dados de emergência
PreencheDadosPess	Preenche os dados pessoais
PreencheDadosEmerg	Preenche os dados de emergência
PreencheDadosFich	Preenche os dados do ficheiro
RetDadosPess	Retorna os dados pessoais
RetDadosEmerg	Retorna os dados de emergência
RetDadosFich	Retorna os dados de um ficheiro
GetNumFich	Retorna o valor do número de ficheiros na base
GetLastKey	Retorna o valor do último índice utilizado no zip
RemoveFich	Remove os dados relativos a um ficheiro
EditaDadosPess	Edita os dados pessoais
EditaFich	Edita os dados do ficheiro
EditaDadosEmerg	Edita os dados de Emergência

Tabela 4 - Funções exportadas pela Biblioteca de manipulação da Base de Dados

De seguida passamos a descrever sucintamente cada operação:

CriarBase – Recebe um caminho para um directório e o nome do ficheiro que conterá base de dados a criar. Cria a base de dados, as suas respectivas tabelas e *triggers*.

CriarBaseEmer – Semelhante à CriarBase, mas cria um ficheiro de base de dados com a tabela relativa aos dados de emergência.

PreencheDadosPess, PreencheDadosEmerg, PreencheDadosFich – Recebem o caminho para o ficheiro base de dados correspondente e os dados a preencher nas tabelas por intermédio de *strings*. Os dados são então convertidos em instruções SQL e enviados para as respectivas tabelas na base de dados.

RetDadosPess, RetDadosEmerg – Recebem o caminho para o ficheiro que contém a base de dados correspondente e um ponteiro para uma estrutura de dados. Tratam o resultado do pedido por instrução SQL à base de dados convertendo as informações necessárias em *strings*. Finalmente preenchem a estrutura de dados apontada com as essas informações.

RetDadosFich – Semelhante à RetDadosPess e RetDadosEmerg mas recebendo ainda um índice de determinado ficheiro. Após o mesmo tratamento de informação a estrutura de dados apontada é também preenchida com a informação do ficheiro a que corresponde o índice.

GetNumFich – Recebe o caminho para o ficheiro que contém a base de dados e retorna o número total de entradas na tabela de ficheiros.

GetLastKey – Recebe o caminho para o ficheiro base de dados e retorna o campo "índex" do último ficheiro inserido na tabela de ficheiros.

RemoveFich – Recebe o caminho para o ficheiro base de dados e valor de um índex, remove a entrada na tabela de ficheiros correspondente ao valor de índex recebido.

EditaDadosPess, EditaDadosEmerg – Recebem o caminho para o ficheiro que contém a respectiva base de dados e os dados a editar sob a forma de *strings*. Após a conversão em instruções SQL actualiza os dados nas respectivas tabelas das bases de dados.

EditaFich – Semelhante às EditaDadosPess, EditaDadosEmerg, mas recebendo também um valor de um índex. Após o mesmo tipo de conversões actualiza as entradas na tabela de ficheiros da base de dados.

A base de dados do protótipo de pEHR está distribuída por dois ficheiros distintos. O primeiro possui duas tabelas, a tabela de dados pessoais e a tabela relativa às informações dos ficheiros clínicos. Os dados de emergência pela sua já referida característica diferenciada em termos de acesso, estão alojados num outro ficheiro de base de dados, onde apenas uma tabela existe, a dos dados de emergência do utilizador.

Embora exista esta separação física em dois ficheiros diferenciados, a interacção e manipulação de dados em ambos é processada da mesma forma e sem nenhum tipo de incompatibilidade.

A título demonstrativo algumas informações foram incluídas nas tabelas da base de dados, dado o carácter demonstrativo da aplicação prova de conceito. Os conteúdos de cada tabela da base de dados são apresentados na Tabela 5.

Dados Pessoais
Nome
Sexo
Data e Nascimento
Estado Civil
Grupo Sanguíneo
Morada

Dados de Emergência
Alergias
Alergias a Medicamentos
Precauções
Historial cirúrgico
Outros

Dados do Ficheiro Clínico
Tipo
Sub-Tipo
Data de Realização
Data de Introdução
Clínico Responsavel
Comentários
Índex do Arquivo Zip

Tabela 5 - Conteúdos das tabelas da base de dados

A tabela de Dados de Emergência na base de dados conjuga as informações dos Dados Pessoais e dos Dados de Emergência que se encontram descritos na Tabela 5. A tabela de dados do ficheiro clínico é composta por várias linhas que partilham os seus atributos, representando cada uma delas um ficheiro introduzido no arquivo.

Realça-se o facto de os campos/tabelas escolhidos para o protótipo não terem sido alvo de estudo de normas. Isto é, o objectivo não era desenvolver um sistema de informação /modelo de dados mas uma arquitectura de pEHR "*portable*". O sistema de informação é minimalista apenas para suportar a prova de conceito.

É de destacar na implementação da base de dados a inclusão de procedimentos do tipo *trigger*. O *trigger* é um tipo de procedimento especial da linguagem SQL que se executa automaticamente sempre que determinado evento ocorra na base de dados.

Tal como referido na secção 4.3.1, sempre que um ficheiro é apagado no arquivo zip, o seu valor de indexação é actualizado. Valor este que é guardado na base de dados aquando da introdução do ficheiro no arquivo zip. Porém este valor não se mantém constante, ou seja, não representa sempre o mesmo ficheiro. Desta forma é necessário reproduzir na base de dados este processo que a estrutura do zip realiza de forma automática. Por exemplo quando ocorre a eliminação de um exame ou relatório médico, o seu ficheiro de dados é apagado do arquivo e a sua entrada na base de dados é eliminada. A partir desse momento é activado o *trigger* implementado que actualiza o campo "índex do zip" de cada entrada na base de dados que possua índex superior. Assim garantimos que as entradas da base de dados se mantêm correctamente ligadas aos seus ficheiros correspondentes.

Outro procedimento automatizado foi implementado com o objectivo de preencher o campo "Data de Introdução" com a data actual, sendo activado sempre que um novo ficheiro seja introduzido no arquivo.

O retorno da informação armazenada nas tabelas da base de dados é processado de forma singular pela nossa aplicação. Como usual na linguagem SQL quando se pretende este retorno, são usados um conjunto de "perguntas" (*querys*) através do comando "*SELECT*", onde se seleccionam dados de uma determinada tabela. Por exemplo para retornar a tabela inteira de dados pessoais é utilizado o comando SQL "*SELECT * from DadosPessoais*", que retorna todas as entradas da tabela de dados pessoais. Estes dados são então retornados pela biblioteca SQLite sob a forma de um *array* de ponteiros para *arrays* de caracteres. Cada um destes ponteiros aponta para um *array* de caracteres onde são guardadas os dados dessa determinada tabela de forma sequencial. Os primeiros *array*'s são preenchidos com os nomes dos parâmetros da tabela (nome, sexo,...) e os seguintes com os seus respectivos valores (Rui Botte, masculino,...). Trata-se de um mecanismo muito simples e algo rudimentar, face às soluções actuais de manipulação de bases de dados relacionais, mas que permite operar em modo "*portable*".

Assim, foi necessário adaptar esta informação de forma a que fosse interpretada da melhor forma pela interface gráfica. Para tal foi criado um processo de tratamento e ligação dos dados entre a biblioteca de interacção com a base de dados (escrita na linguagem C) e a interface de visualização (desenvolvida em Visual Basic).

Para que se compreenda melhor este processo de transferência de informação no esquema da Figura 13 é dado o exemplo do retorno da informação pessoal do paciente, da base de dados para a interface gráfica de visualização.

O processo baseia-se inicialmente em capturar quais os *arrays* de caracteres que correspondem aos valores dos campos da tabela e ignorar os que armazenam o nome de campo. Estas informações (campos da tabela) vão ser copiadas para uma estrutura de dados que possui membros equivalentes aos que a base de dados possui. Uma estrutura de dados idêntica existe na interface gráfica (Visual Basic) e é transmitido um ponteiro, dessa estrutura, para a biblioteca (dll) através da função *RetDadosPess* já citada. Esta função responsabiliza-se por carregar a informação na estrutura da base de dados, através do seu ponteiro, com a informação que se encontra na estrutura análoga da biblioteca (C). A interface gráfica possui a

partir desse momento a informação que necessita na sua estrutura de dados, a partir daí pode usá-la para a apresentar ao utilizador.

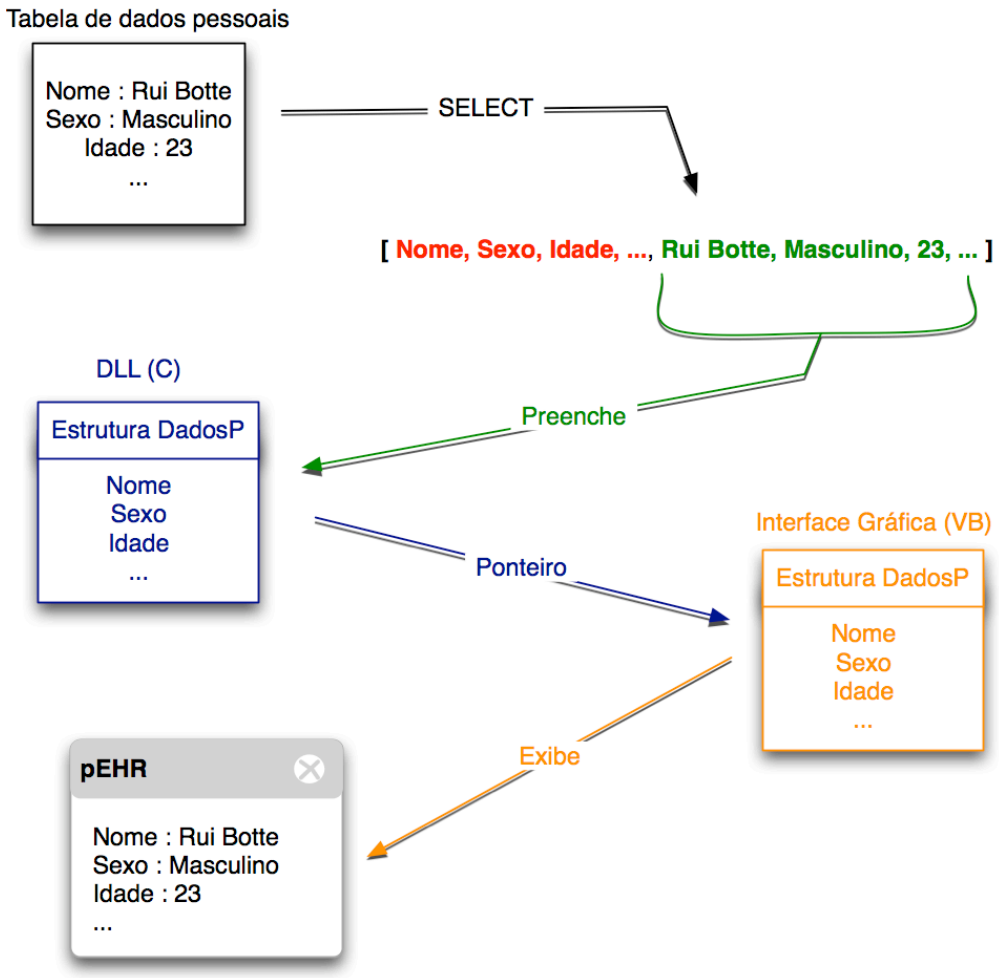

Figura 13 - Retorno da informação da base de dados para a interface gráfica

Como a tabela de informação pessoal apenas contém uma entrada (linha), o processo é realizado uma única vez. Tal facto é exactamente igual na tabela relativa aos dados de emergência. O retorno da informação dos ficheiros no arquivo é semelhante, com a ressalva que o acesso é repetido para cada linha da tabela, pois esta possui tantas linhas quanto o número de ficheiros que tenham sido inseridos no arquivo. Desta forma em cada uma das iterações ao longo de toda a tabela, a estrutura de dados é sucessivamente carregada com nova informação de cada linha e exibida pela interface gráfica, tal como no exemplo anterior.

4.3.3 Segurança e Controlo de Acessos

Como foi definido um dos requisitos da nossa aplicação é garantir a segurança da informação assim como controlar o seu acesso. Um dos maiores paradigmas encontrados na implementação de um controlo de acessos eficaz passa por determinar o melhor processo de arquivar em segurança as credenciais que potenciam o acesso. Se num comum sistema de controlo de acessos podemos guardar essas informações num outro local que ofereça melhor segurança, quando falamos em controlar acessos a partir de um dispositivo móvel tudo o que seja necessário guardar terá necessariamente que ficar no dispositivo.

O controlo de acessos da aplicação foi abordado segundo dois diferentes tipos de implementação. A escolha por um destes sistemas de acesso obriga a um compromisso entre as suas vantagens e desvantagens. Dado o carácter demonstrativo da aplicação, ambas as implementações e suas características serão expostas. No final desta secção serão referidas algumas considerações gerais de segurança.

4.3.3.1 Controlo de Acessos com Base num Ficheiro Externo

A primeira abordagem tomada com vista a implementar um sistema de controlo de acessos, foi usar uma política de credenciais de acesso com diferentes utilizadores e passwords. Desta forma foi elaborado um sistema baseado num simples ficheiro de texto que contém as informações de acesso, de forma a ser de fácil leitura pela nossa aplicação. Por outro lado, a sua leitura directa não é possível já que a informação se encontra protegida através do uso de algoritmos de *hashing* e de cifra.

O ficheiro é composto por várias linhas, cada uma delas correspondente a um utilizador e organizada por 3 blocos de texto separados pelo carácter ‘ : ’ (dois pontos). O primeiro bloco é o nome do utilizador, o segundo uma *saltedhash* da password do utilizador e finalmente o terceiro a password do zip cifrada com a password do utilizador. O aspecto de uma destas linhas será algo semelhante ao seguinte:

Utilizador1**:**db634151a02a629975d1454f78fa36343cbea75a405506b82e76b45b1c4b246167eab1b5**:**1123464ee7b6a0c917348e690ba29535

O primeiro bloco corresponde ao nome do utilizador, é usado para comparação com o nome que o utilizador introduz na aplicação. O segundo bloco possui um tamanho fixo de 72 caracteres hexadecimais e pode ser dividido em duas partes distintas. A primeira composta por 64 caracteres hexadecimais (512bits), representa uma *hash* obtida a partir do uso do algoritmo criptográfico de *hashing sha512* sob a password do utilizador à qual se adicionou um valor gerado aleatoriamente de 8 caracteres hexadecimais (64bits). Este último valor constitui a segunda parte do bloco. A este método específico de *hashing* é dado o nome de *Salted Hashing* e a segunda parte do bloco é conhecida por *Salt*, sendo um método muito utilizado em sistemas *Unix* para arquivo de passwords. [41]

Finalmente, o terceiro bloco é composto por 32 caracteres hexadecimais (256bits) a que corresponde a password com que os ficheiros são cifrados no zip. Este bloco é obtido através do método criptográfico *aes256* aplicado sobre a password do zip e usando a password do utilizador.

Apresentam-se na Figura 14 algumas relações esquemáticas e as suas respectivas descrições para que se compreenda da melhor forma o processo de criação do ficheiro de credenciais.

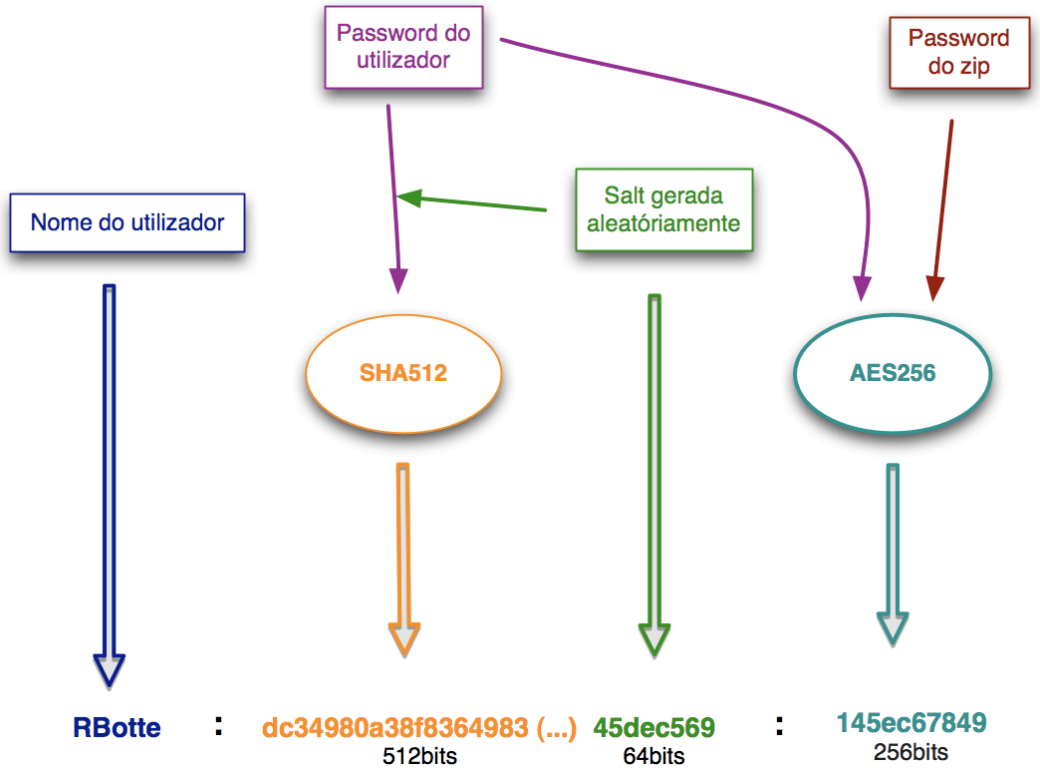

Figura 14 - Exemplo da criação de uma linha do ficheiro de autenticação

O método de validação do utilizador/password funciona praticamente de forma inversa ao processo de criação do ficheiro. O processo de validação descrito na Figura 15 é executado da seguinte forma: O utilizador introduz o seu nome e password e a primeira etapa passa por verificar se no ficheiro se encontra um utilizador com esse nome. Seguidamente assumindo que o utilizador existe, retiram-se os 64bits da *salt* e juntamente com a password inserida gera-se uma *hash* pelo mesmo algoritmo *sha512*. Esta *hash* gerada é comparada com a que está no ficheiro e se corresponder é concedido ao utilizador o acesso à aplicação. Para aceder à informação a aplicação necessita da password do arquivo zip, a qual é decifrada a partir do terceiro bloco da linha, usando a password do utilizador que já foi anteriormente validada.

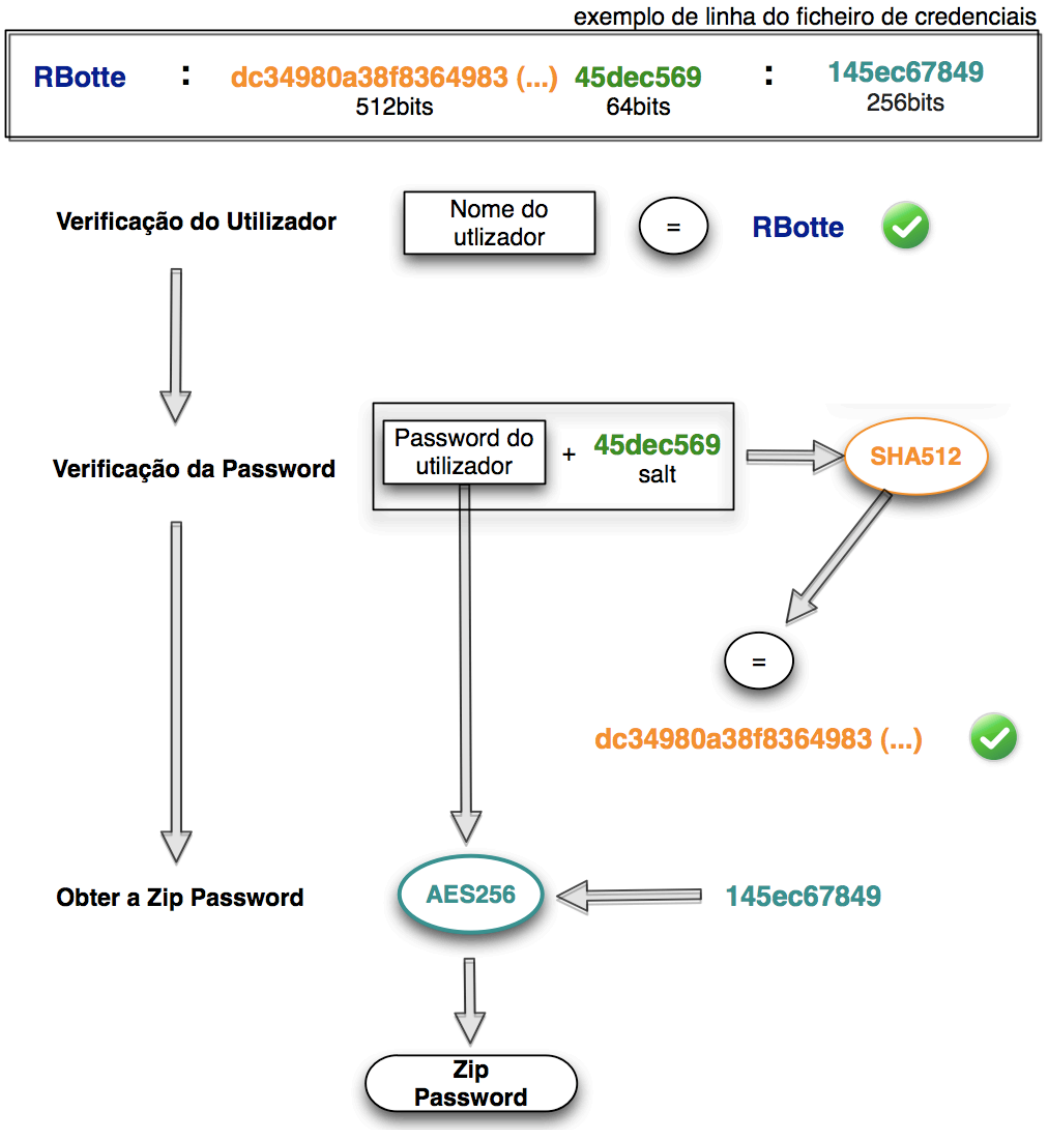

Figura 15 - Diagrama de validação do utilizador e obtenção da password do zip

Desta forma e durante a corrente sessão, através da password, a aplicação tem a possibilidade de aceder aos ficheiros protegidos no arquivo zip.

Efectuando o controlo de acessos com recurso a um ficheiro externo de autenticação, a sua respectiva biblioteca de interacção é utilizada, tal como acontece nos casos anteriores com o arquivo zip e a base de dados. Na biblioteca em questão algumas funções foram implementadas. Destas, as que são exportadas de modo a possibilitar o seu uso através da interface gráfica, assim como sua respectiva finalidade, são listadas na Tabela 6.

Nome da Função	**Finalidade**
AuthUser	Valida a autenticação do utilizador
GenerateHash	Gera uma *Hash* a partir de uma *string*
GenRndStr	Gera uma *string* aleatória
EncriptarPw	Codifica uma Password
DesencriptarPw	Descodifica a Password
MakeShadowFile	Cria o ficheiro de Autenticação

Tabela 6 - Funções exportadas pela biblioteca de controlo de acessos

De seguida passamos a descrever sucintamente cada operação:

AuthUser – Recebe um nome de utilizador e uma password, verifica no ficheiro de autenticação a existência desse utilizador, assim como, se a sua password está correcta. Responde com um valor booleano verdadeiro ou falso, conforme o utilizador e a password sejam ou não validados.

GenerateHash – Recebe a password do utilizador e gera através do algoritmo criptográfico a *hash* correspondente à password de utilizador.

GenRndStr – Retorna uma *string* gerada aleatoriamente, que será usada como password de protecção dos ficheiros no arquivo zip.

EncriptarPw – Recebe a password do utilizador e a password dos ficheiros no zip gerada aleatoriamente e através do algoritmo criptográfico cifra a password dos ficheiros com a password do utilizador, retornando a correspondente password cifrada.

DesencriptarPw – Recebe a password dos ficheiros no arquivo zip cifrada e a password do utilizador, através desta decifra a password dos ficheiros no arquivo zip e retorna-a.

MakeShadowFile – Recebe o nome do utilizador a *hash* da password do utilizador e a cifra da password dos ficheiros no zip, criando o ficheiro que contém essas informações.

Este método para controlo de acessos, baseado num ficheiro externo à aplicação, tem por principal característica permitir uma maior flexibilidade, isto é, permitir adicionar e remover vários utilizadores com diferentes passwords de acesso. Porém, tendo em conta que o pEHR é uma aplicação centrada no seu portador, permitir o acesso indiferenciado a diversos utilizadores é uma funcionalidade que não traz vantagens relevantes à sua utilização.

Muito embora o algoritmo de codificação da password do zip seja robusto, está dependente unicamente da password do utilizador, que ao ser comprometida permite o acesso directo à password do zip, ou seja, neste caso a eventual robustez da password do zip (gerada aleatoriamente) torna-se irrelevante.

Por outro lado, salienta-se como sendo uma vantagem, o facto deste sistema permitir a alteração das credenciais de acesso, algo que o segundo tipo de implementação não permite (Secção 4.3.3.2).

4.3.3.2 Controlo de Acessos via Zip Password

Uma diferente abordagem ao controlo de acessos foi efectuada com vista não só a simplificar este processo como também a melhorar a sua robustez. A complexidade e as possibilidades multi-utilizador da implementação anterior, tornam-se excessivas para a tentativa de demonstração de conceito que apresentamos. Deste modo foi implementado um controlo alternativo mais simples de acesso para garantir um mínimo razoável de segurança no acesso à aplicação.

Evitando o recurso a uma biblioteca extra e o uso de um ficheiro externo onde guardar credenciais de acesso, usamos a password de acesso ao ficheiros no zip como forma de

validação alternativa do utilizador. O utilizador na primeira utilização da aplicação insere a sua password pessoal e esta é utilizada para criar a password dos ficheiros no zip. Optou-se por não a utilizar directamente mas sim anexar-lhe um conjunto de caracteres e algarismos (32) definidos internamente na aplicação para aumentar a sua robustez. Resumindo, a password do zip é composta por duas componentes, a password do utilizador e uma definida internamente.

Este método de anexação de dois diferentes componentes para criação da password que protege os arquivos no zip possibilita que, a password que o utilizador utiliza para acesso à aplicação não seja a mesma que permitirá o acesso directo aos ficheiros do arquivo zip, que sendo fraca poderia ser facilmente quebrável.

A utilização de uma password "*softcoded*", i.e. definida internamente na aplicação, como complemento à password do utilizador atribui ao arquivo zip uma maior robustez contra ataques de "força bruta". Se a password de acesso ao zip fosse definida apenas internamente, mesmo que robusta, ao ser comprometida todos os dispositivo pEHR seriam também comprometidos.

O acesso à aplicação estará sempre dependente das credenciais que o utilizador escolhe, pelo que para minimizar essa vulnerabilidade, a aplicação poderá forçar o uso de uma password mais imune a "ataques" (uso de um mínimo de algarismos e caracteres especiais p.e.).

O modo de autenticação deste modelo é bastante simples, necessitando apenas replicar inversamente o processo de criação da password do zip. O utilizador insere a sua password pessoal, esta é anexada à *string* definida internamente e enviada para o arquivo zip seguida de uma instrução de acesso a um ficheiro. Seguidamente o arquivo zip ao receber essa instrução tenta validar a password para aceder ao ficheiro, se a password estiver incorrecta é gerada uma interrupção específica que é capturada pela nossa aplicação e invalida o acesso do utilizador.

No caso em que a password inserida pelo utilizador seja a correcta, nenhuma excepção é gerada e é validado o acesso ao utilizador (a instrução enviada ao zip é apenas um tentativa de edição de um ficheiro sem nenhum parâmetro pelo que nada acontece ao arquivo). O diagrama da Figura 16 sintetiza o processo supramencionado.

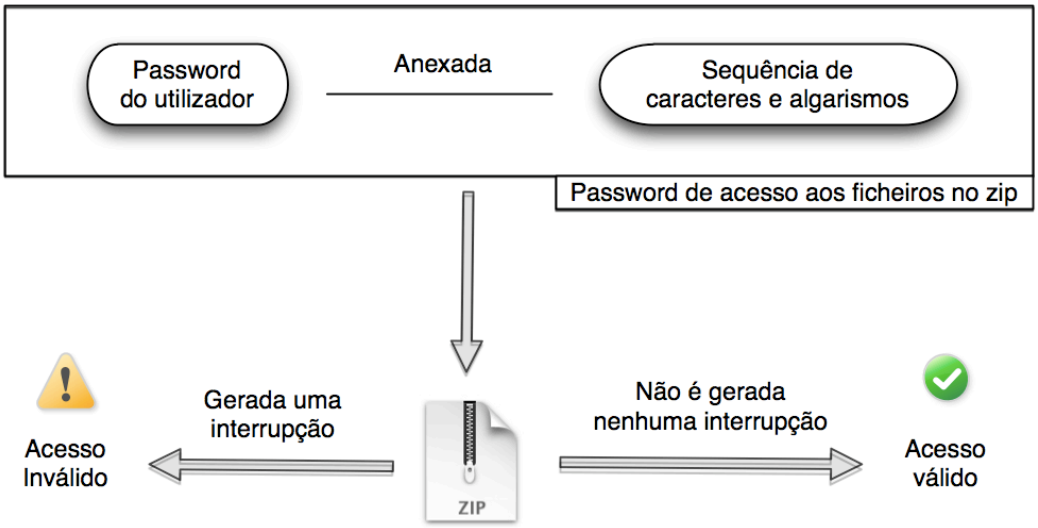

Figura 16 - Diagrama de validação do utilizador usando o arquivo zip

Este tipo de controlo de acesso, suportado pelo sistema de validação da password dos ficheiros no arquivo zip, é um sistema de implementação mais simples que o anterior (4.3.3.1) já que não necessita de utilizar um ficheiro e uma biblioteca extra. Porém, a alteração da password dos ficheiros no zip não é possível, daí que as credenciais de acesso à aplicação também não possam ser alteradas. Desta forma, o utilizador terá de manter a mesma password de acesso durante toda a utilização do pEHR.

4.3.3.3 Considerações Gerais de Segurança

A inclusão dos ficheiros clínicos e de base de dados no arquivo zip, permite preservá-los de forma segura, mesmo assim, não é capaz de garantir 100% de segurança, pois existe sempre a possibilidade de desenvolver uma forma de "quebrar" uma codificação. Os algoritmos de segurança utilizados pelo arquivo zip (ex. AES256), garantem no entanto, um mínimo aceitável de protecção contra esse tipo de ataques.

Os ficheiros que contêm a base de dados estão no interior do arquivo zip e são extraídos para o sistema em *run-time* i.e. durante a execução da aplicação, e no final da utilização os ficheiros actualizados são repostos no interior do arquivo zip e eliminados do

sistema de ficheiros. Esta necessidade do ficheiro de base de dados ser extraído, advém do facto da biblioteca que interage com esse ficheiro necessitar de um caminho válido no sistema de ficheiros, para lhe poder aceder correctamente.

Os ficheiros de base de dados não possuem nenhum sistema de controlo ao seu acesso, este facto torna vulneráveis as informações que a base de dados possui, durante a execução da aplicação. Porém este processo de extracção e reintrodução no arquivo zip permite, na eventualidade da perca do dispositivo, que nenhuma informação esteja directamente exposta e acessível, já que todos os ficheiros críticos se encontram no interior do arquivo zip

A vulnerabilidade da base de dados durante a execução da aplicação poderia ser colmatada com o uso da licença comercial da biblioteca *SQLite*. Esta versão ao contrário daquela a que tivemos acesso, possibilita a codificação da informação além de permitir a sua manipulação. [42] Utilizando-a adicionaria mais um nível de segurança à aplicação e consequentemente à informação arquivada.

O controlo diferenciado de acessos, como proposto no modelo da secção 3.4 não foi possível de implementar. Em ambos os modelos de controlo de acessos propostos nas secções anteriores (4.3.3.1 e 4.3.3.2) apenas é controlado o acesso principal à aplicação, sem a definição de nenhum tipo de privilégio. Porém, usando a versão comercial do motor de base de dados (*SQLite*), outras credenciais de acesso poderiam ser arquivadas seguramente na base de dados. Essas novas credenciais sim, poderiam permitir um acesso com níveis/privilégios diferenciados à aplicação.

4.3.4 Interface Gráfica

A interface gráfica da aplicação funciona como um meio de interacção, para que o utilizador consiga tirar partido das funcionalidades implementadas nas bibliotecas externas. Para que esta situação ocorra correctamente foi necessário incluir algumas características e funções internas que permitem que esta funcione correctamente e melhore essa interacção.

A interface gráfica possui declarados os protótipos das funções que se localizam nas bibliotecas que compõem a aplicação, assim como, um conjunto de variáveis importantes. Variáveis estas que, durante a execução da aplicação guardam os caminhos para os ficheiros

com os quais a aplicação interage: o arquivo zip, o ficheiro que contém a base de dados e eventualmente o ficheiro que contém as credenciais de acesso. Possui também definidas as estruturas de dados que permitem, pelo método indicado na secção 4.3.2 e Figura 13, o retorno da informação da base de dados.

Na secção 4.3.2 é feita a alusão ao acesso cíclico à informação da tabela da base de dados que contém a informação dos ficheiros. Esta rotina de acesso cíclico a cada linha da base de dados encontra-se também implementada na interface gráfica. A informação da tabela da base de dados é retornada para a interface gráfica linha a linha e esta necessita de ser guardada em algum local, já que a estrutura de dados usada é actualizada com nova informação por cada acesso uma linha da tabela. A cada ciclo, a informação é transferida da estrutura de dados para uma tabela (objecto do tipo *listview* em *Visual Basic 6*) da interface gráfica que a apresenta. Por forma a identificar individualmente cada uma das entradas dessa tabela, é usado como identificador o campo "índex do zip", que não necessita de ser conhecido pelo utilizador mas que é necessário para associar cada entrada ao respectivo ficheiro no arquivo zip.

Na Figura 17 é apresentada a janela principal da aplicação e é possível verificar em destaque as tabelas (objectos *listview*) que apresentam a informação armazenada na base de dados. Na parte superior a tabela relativa aos dados pessoais que apenas contém uma linha, na parte central a tabela relativa às informações dos ficheiros arquivados terá as suas linhas preenchidas pelo método cíclico supracitado.

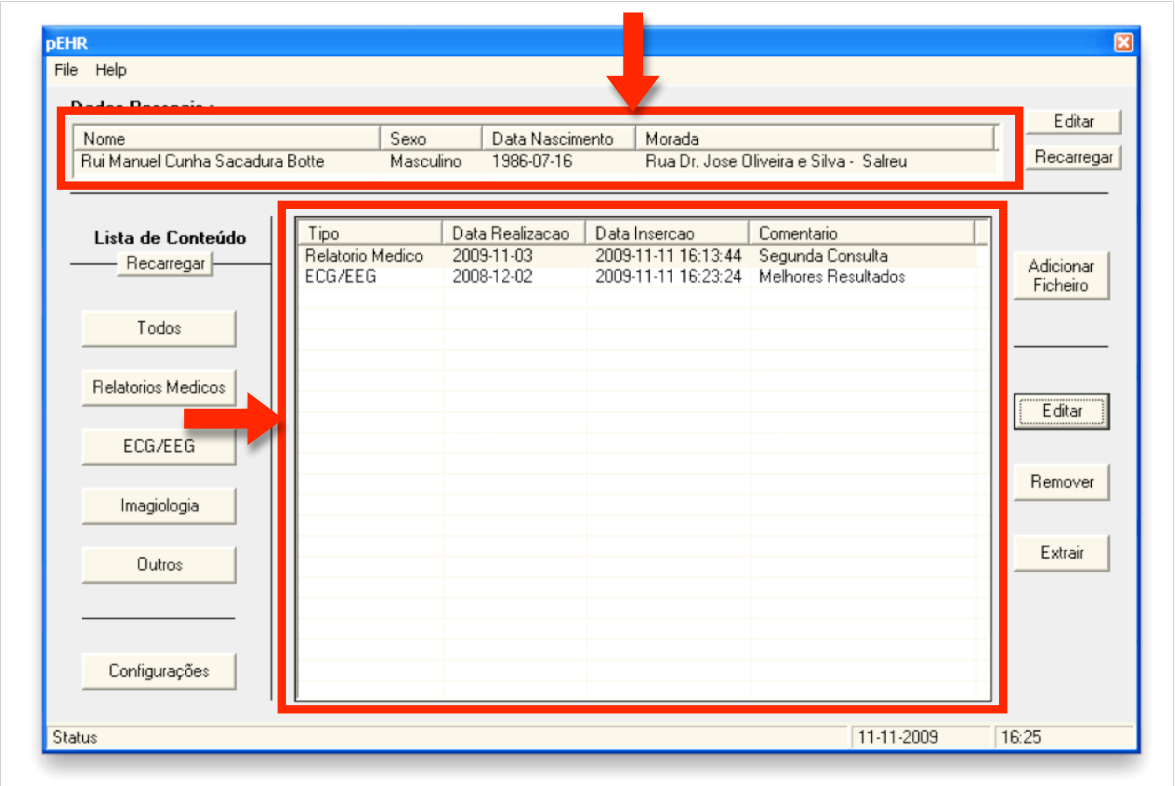

Figura 17 - Destaque das tabelas de informação no ecrã principal da aplicação

4.3.5 Outros Aspectos

O limite de armazenamento da aplicação é virtualmente inexistente, já que está associado ao espaço de armazenamento disponível no dispositivo onde a aplicação se encontra. Porém o sistema de ficheiros em que o dispositivo se encontra formatado limita o tamanho máximo que um ficheiro pode ter. Esta aplicação está orientada para o uso em sistemas *Windows*, sendo os seus principais sistemas de ficheiros o *FAT32* (File Allocation Table 32 bits) e o recente *NTFS* (NT File System). Já que o arquivo *zip* é entendido como um só ficheiro pelo sistema, o seu tamanho máximo num sistema de ficheiros do tipo *FAT32* é de aproximadamente quatro *Gigabytes* [43]; num sistema *NTFS*, este valor aumenta para 16 *Exbibytes* [44] (1 exbibyte = 2^{60} bytes). Desta forma é recomendado, em termos de capacidade, que o dispositivo se encontre formatado segundo o sistema de ficheiros *NTFS*, para que o arquivo zip não tenha o seu tamanho limitado a um baixo valor.

4.4 Demonstrador de pEHR

Para melhor compreensão do funcionamento do protótipo de aplicação desenvolvido, nesta secção será exibida uma sequência de vistas da aplicação. Estas irão demonstrar a sua normal sequência de uso, desde a sua primeira utilização passando pelo seu normal funcionamento até à análise de alguns pormenores de funcionamento.

A primeira vez que o utilizador inicia a aplicação, esta verifica a existência dos ficheiros de arquivo, base de dados e de autenticação (se a aplicação estiver a funcionar segundo este método de controlo de acessos). Tratando-se da primeira vez que a aplicação está ser acedida, esta informa o utilizador desse facto (Figura 18).

Figura 18 - Ecrã da aplicação na primeira utilização

Se algum dos ficheiros não for detectado poderá também significar que algum deles foi corrompido ou apagado e dado que a aplicação necessita de todos para funcionar correctamente, esta terá de ser reiniciada. Em ambas as situações anteriores é sugerido ao utilizador que reinicie/inicie o uso da aplicação (Figura 18). Nesta situação em concreto, será executado um *backup* dos ficheiros que eventualmente ainda existam, colocando no nome do ficheiro a data em que o backup foi efectuado e novos ficheiros são criados como se de uma primeira utilização se trata-se.

De seguida é pedido ao utilizador que configure as suas credenciais de acesso à aplicação (Figura 19 e Figura 20).

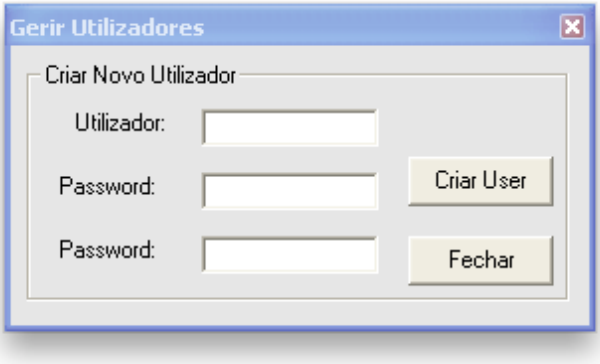

Figura 19 - Painel de gestão de utilizadores

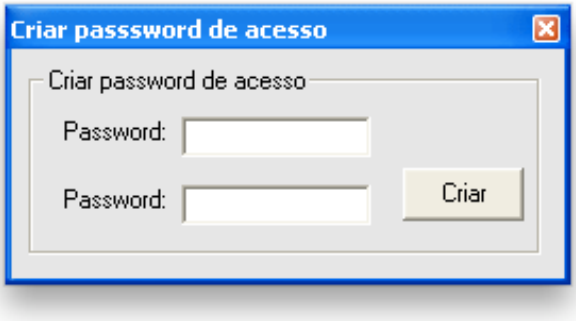

Figura 20 - Painel de criação da password de acesso

Conforme o tipo de implementação usado para o controlo de acessos, o utilizador poderá ter de inserir um nome de utilizador e uma password (Figura 19) ou apenas uma password (Figura 20).

Após o processo de definição de acessos, é solicitada ao utilizador a inserção dos seus dados pessoais e das suas informações em caso de emergência. (Figura 21)

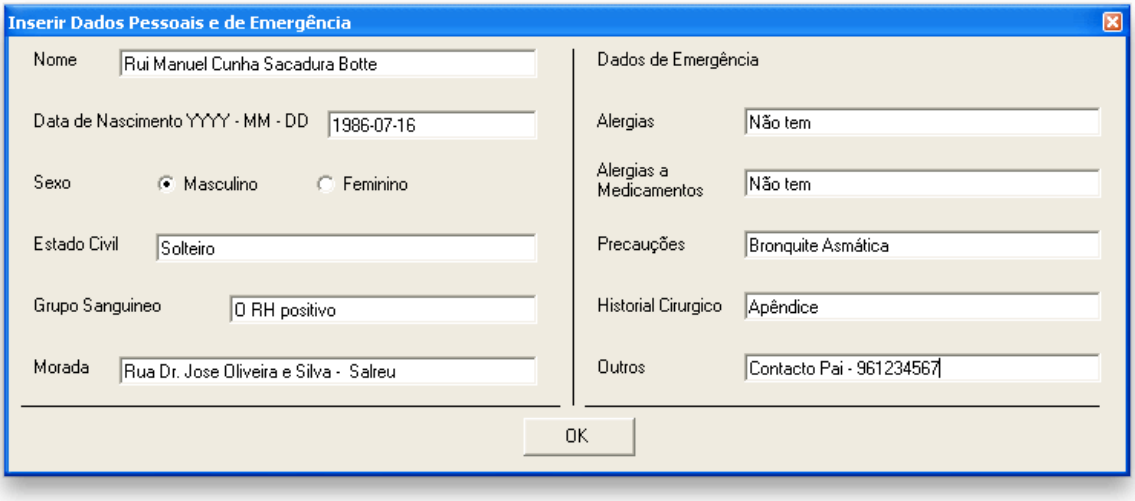

Figura 21 - Inserção dos dados pessoais e de emergência na aplicação

Neste momento a aplicação já possui as principais informações de que necessita para funcionar normalmente. Aparece então no ecrã a janela inicial da aplicação, janela esta, que surgirá a partir deste momento e sempre que o utilizador a voltar a iniciar. (Figura 22)

Figura 22 - Ecrã inicial do pEHR após a primeira utilização

Verifica-se que a janela inicial (Figura 22) possui três botões de interacção distintos que permitem respectivamente: o acesso aos dados de emergência, a autenticação do utilizador para acesso à restante aplicação ("Login") e finalmente um que permite abandonar o seu uso ("Fechar").

Acedendo ao botão "Dados de Emergência" é exibida uma janela (Figura 23) onde são apresentados os dados de emergência inseridos pelo utilizador. Desta forma permite-se uma consulta rápida por parte dos responsáveis pelo apoio de urgência, numa situação em que o detentor do *pEHR*, possa estar impossibilitado de as fornecer. Mais uma vez, chama-se à atenção que os elementos de informação são meramente demonstrativos.

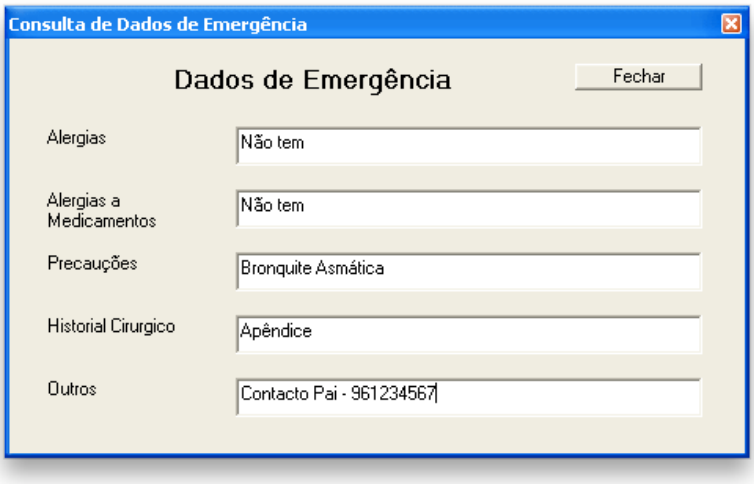

Figura 23 - Consulta de Dados de Emergência

Para aceder ao restante conteúdo da aplicação é necessário na janela inicial (Figura 22) aceder ao botão "Login", surgirá então uma tela onde o utilizador pode inserir as suas credenciais de acesso. Mais uma vez conforme o tipo de controlo de acessos implementado duas diferentes telas poderão surgir. (Figura 24) (Figura 25)

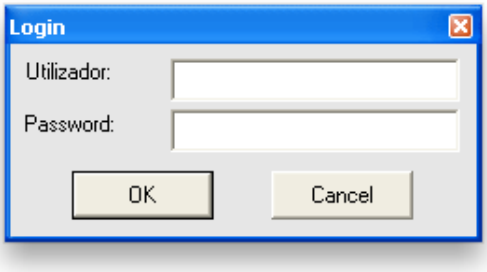

Figura 24 - Janela de autenticação com nome de utilizador e password

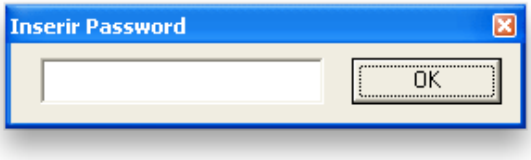

Figura 25 – Janela de autenticação apenas por password

Após a introdução das credenciais de acesso e se estas forem validadas correctamente, será permitido ao utilizador aceder ao ecrã principal da aplicação (Figura 26).

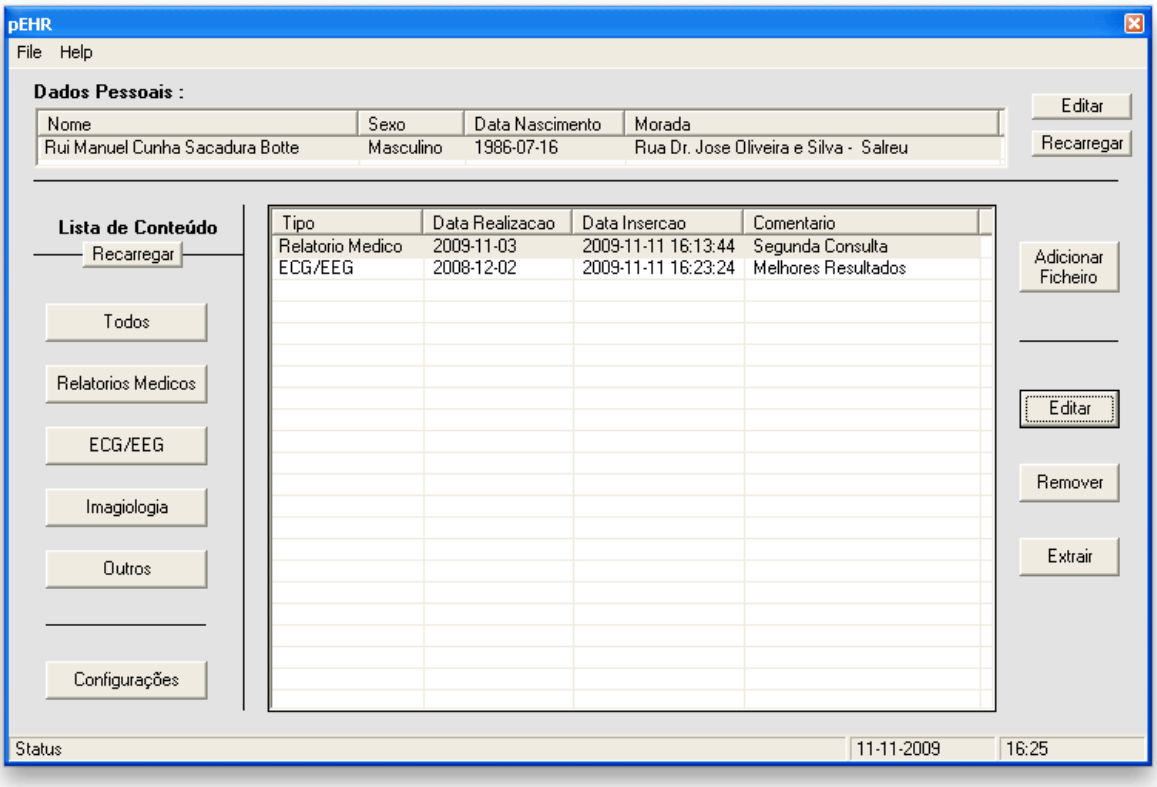

Figura 26 - Janela principal da aplicação já com alguns ficheiros inseridos

Verifica-se na Figura 26 a existência de uma tabela principal, ao centro, onde é possível visualizar uma listagem de todos os ficheiros inseridos e as suas informações associadas.

O botão, localizado à direita da tabela na parte superior "Adicionar Ficheiro" é possível seleccionar um ficheiro para adicionar ao pEHR (Figura 27), seguido da janela onde é possível inserir as suas informações (Figura 28). Após seleccionar alguma entrada da tabela é possível interagir com o ficheiro respectivo, usando os botões laterais, editando informações (botão "editar") (Figura 29) ou eliminá-lo (botão "eliminar").

Na parte superior do ecrã encontramos os principais dados pessoais do utilizador e clicando no botão à sua direita é possível visualizar, assim como, editar estas informações (Figura 30) .

É possível também verificar no ecrã principal (Figura 26) a existência de dois botões "recarregar" junto dos dados pessoais e da lista de conteúdo. Botões estes, que permitem forçar uma actualização de ambas as informações.

Os botões lateralmente à esquerda da tabela central (Figura 26), permitem uma filtragem da listagem da tabela, mediante o botão selecionado, por tipo de ficheiro: "Todos", "Relatórios Médicos", "ECG/EEG", "Imagiologia" e "Outros".

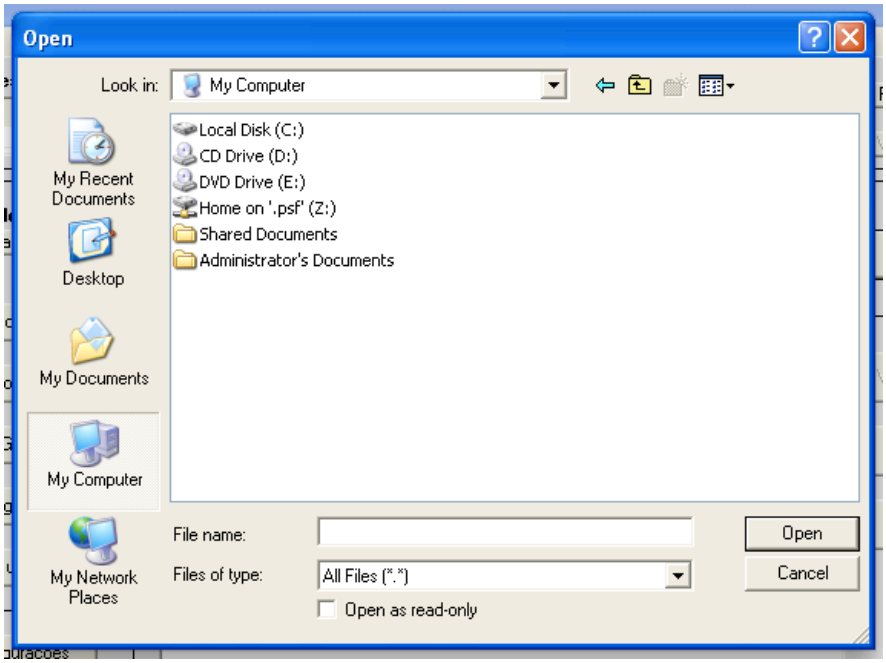

Figura 27 - Janela de selecção do ficheiro a adicionar

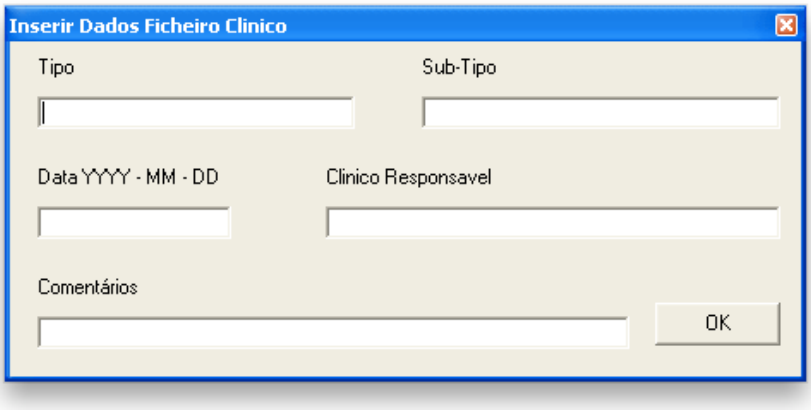

Figura 28 - Janela de inserção dos dados relativos ao ficheiro clínico

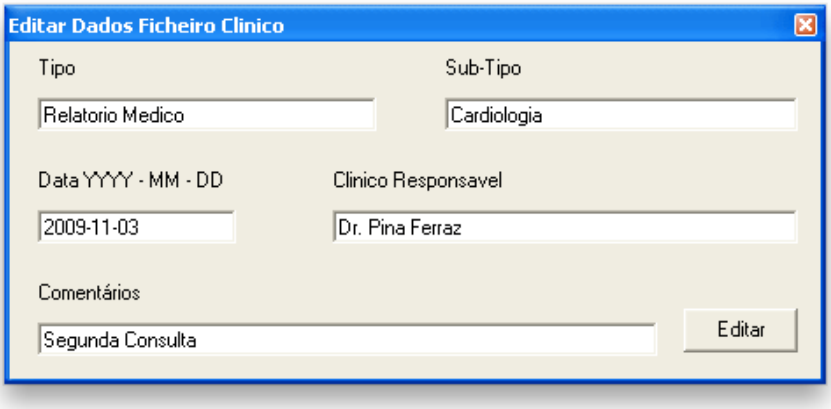

Figura 29 - Janela de edição dos dados do ficheiro clínico

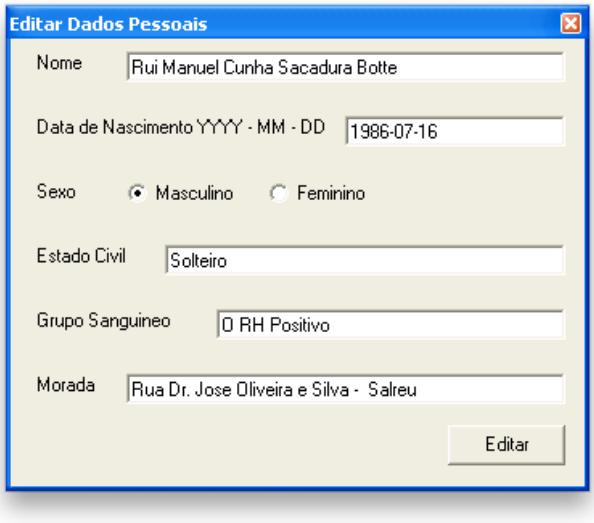

Figura 30 - Janela de Edição dos dados pessoais

Finalmente, e de modo a permitir a visualização do ficheiro que pretendemos o botão "Extrair" (Figura 26), permite a extracção do ficheiro seleccionado na tabela para um local em disco a seleccionar (Figura 31).

Figura 31 - Janela de selecção de destino para extracção do ficheiro

Esta modalidade de extracção, funciona como demonstração da funcionalidade de visualização do ficheiro, obrigando o utilizador a extrair e abrir cada ficheiro que pretenda consultar. Para uma situação de visualização directa seria necessária a inclusão de um visualizador interno compatível com a extracção para memória do arquivo zip.

Capítulo 5 - Conclusões e Trabalho Futuro

Neste capítulo pretende-se compor um pequeno resumo e conclusão do trabalho realizado ao longo deste estudo, bem como, referir algumas linhas possíveis para continuação e melhoramento do mesmo.

5.1 Resumo e Conclusões

O principal intuito desta dissertação, prende-se com o estudo e experimentação de diversas tecnologias, as quais, tornaram possível o desenvolvimento de uma plataforma de acesso integrado a informação complementar de diagnóstico.

Com o advento das plataformas de desenvolvimento baseadas em Java VM ou no .Net *Framework,* surge uma necessidade de dependência que, de certa forma, ignora a utilização das aplicações desenvolvidas em diferentes máquinas, pois pressupõe que os *frameworks* ou *Virtual Machines* se encontram pré-instalados ou disponíveis para instalação. A instalação quer da aplicação quer de outro tipo de plataforma que permita o seu funcionamento é uma realidade que se pretende evitar ao desenvolver uma aplicação de características portáteis, como é o caso deste estudo.

A escolha de ferramentas a utilizar no desenvolvimento desta dissertação passou por diversas fases que incluíram o seu estudo e experimentação. Ao invés de controlar as dependências da aplicação no sistema através de processos de virtualização, ou reutilizar aplicações já desenvolvidas, o processo de experiência culminou numa decisão final de

desenvolvimento integral da aplicação. No seguimento deste diferente rumo adoptado, foi tido em conta como principal objectivo evitar o recurso e dependência de terceiras plataformas. Desta forma, recorrendo às ferramentas adoptadas foram desenvolvidas as principais funcionalidades propostas, assim como, uma interface gráfica demonstrativa.

Este trabalho não teve por principal objectivo desenvolver um completo sistema de informação clínica, mas sim um meio de importação de objectos persistentes, (incluindo relatórios, documentos imagiológicos, sinais electrocardiográficos, etc.) que se apresentem sob o formato digital. Assim como, outro tipo de documentos que o utilizador considere relevantes e que possa digitalizar, por forma a incluí-los no seu registo médico transportável.

Com o trabalho realizado conclui-se que a exequibilidade do conceito de registo médico pessoal transportável é possível, mesmo que, usando ferramentas que actualmente deixaram de ser as mais utilizadas face ao surgimento de novas plataformas de desenvolvimento. Foi desenvolvida uma aplicação pEHR prova de conceito de uma forma original, simples e usando tecnologias bastante comuns de arquivo e compressão de ficheiros (arquivo *zip*), juntamente com um repositório de dados elementar como é o caso do motor de base de dados *SQLite*.

Muito embora a sua implementação não seja completa, seria de certa forma impossível poder dar por totalmente concluído um trabalho desta natureza. A aplicação desenvolvida como demonstrador de funcionalidades consegue provar que um registo médico desta índole é implementável. Da mesma forma que os principais custos para o cidadão estão basicamente associados ao suporte móvel do pEHR. A sua independência face ao dispositivo onde se encontra garante no futuro a sua expansibilidade quer em tamanho de armazenamento, quer em velocidade de acesso.

Este trabalho proporcionou a angariação de conhecimentos em diferentes áreas ao permitir o contacto e experimentação com diversos tipos de tecnologias, constituindo uma importante e gratificante mais valia pessoal.

Apresenta também, um suporte e uma linha de continuidade para um possível desenvolvimento futuro. Novas características poderão ser acrescentadas a esta aplicação de modo a aumentar suas funcionalidades e conduzir a um acréscimo da sua qualidade e usabilidade, como será apresentado na seguinte secção (5.2).

5.2 Trabalho Futuro

Este tipo de estudo estará sempre dependente das tecnologias emergentes que poderão favorecer ou prejudicar a aplicação baseada neste modelo. Porém, baseando-se apenas no trabalho desenvolvido, para concluir esta dissertação é necessário referir ainda algumas possibilidades de melhoramento futuro.

A aplicação desenvolvida proporciona uma interacção simples e uma demonstração de funcionalidades básicas de arquivo e visualização de informação, carecendo de outras funcionalidades que a beneficiariam. Obviamente, seria impossível referir ou analisar todas elas, deste modo, cingimo-nos a analisar aquelas que se consideram mais pertinentes.

Uma das principais características a incluir na aplicação será a do suporte aos standards de portabilidade clínica referidos no estado da arte desta dissertação (CDA, DICOM, etc). Desta forma seria possível, no caso de uma fonte de dados compatível com alguns dos standards, a sua integração directa no registo. As informações relativas ao ficheiro seriam adquiridas de forma automática pela aplicação, evitando o seu preenchimento manual pelo utilizador, bem como armazenar na base de dados outras características relevantes mediante o tipo de fonte (ex. DICOM dados para indexação).

O controlo de acessos pode ser aperfeiçoado, criando diferentes perfis de utilização. Relacionados não só com as permissões de leitura ou edição da informação, mas também com a filtragem por tipo ou especialidade. Controlando desta forma, quem e a que informação pode ser acedida. Isto será possível, atribuindo no arquivo zip múltiplas passwords a diferentes conjuntos de ficheiros e utilizando o módulo de codificação da base de dados para as conter de forma segura.

Outra funcionalidade a incluir na aplicação seria um módulo de importação directa de informação através de um dispositivo de digitalização (ex. *scanner*). O utilizador poderia importar documentos que se encontrem em formato analógico sob a forma de imagem ou documento electrónico, PDF (Portable Document Format), por exemplo. Com certeza não com todo o potencial dos formatos standard electrónicos, mas certamente como uma alternativa segura e organizada de os arquivar.

Não existindo forma de impedir a destruição da informação do dispositivo pEHR, para prevenir a sua perda total, poderia ser implementada uma funcionalidade de *backup* (cópia de segurança). Através desta, seria possível salvaguardar a informação em outra localização (ex. computador pessoal) e permitir posteriormente o seu restauro.

A extracção dos ficheiros para disco para visualização tem um propósito demonstrativo e menos seguro. Este último aspecto seria melhorado com a inclusão na aplicação de um visualizador próprio ou um externo, com a capacidade de consumo de informação extraída para memória.

Finalmente para poder beneficiar um maior número de utilizadores, a aplicação poderia ser portada para outros sistemas operativos (Mac OS, Linux). Grande parte do código desenvolvido poderá ser reutilizado, assim como as bibliotecas utilizadas (*ZipArchive* e *SQLite*) que se encontram disponíveis para esses sistemas operativos.

Bibliografia

1. HALAMAKA, J.; MANDL, K. D.; TANG, P. C. Early Experiences with Personal Health Records. **Journal of the American Medical Informatics Association**, v. 15, n. 1, Feb 2008.

2. W D BIDGOOD, J.; HORII, S. C. Introduction to the ACR-NEMA DICOM standard. **Radiographics**, p. 345-355, March 1992.

3. PLOTNIKOV, V. A.; PRILUTSKII, D. A.; SELISHCHEV, S. V. The SCP-ECG standard in electrocardiographic software systems. **Biomedical Engineering**, v. 33, n. 3, p. 128-135, Maio 1999.

4. DOLIN, R. H. et al. HL7 clinical Document Architecture, Release2. **Journal of the American Informatics Association**, p. 30-38, September 2005.

5. ELBERG, P. B. Electronic patient records and innovation in health care services. **International Journal of Medical Informatics**, n. 64, p. 201-205, 2001.

6. GOOGLE. About Google Health. **Google Health**, 2008. Disponivel em: <http://www.google.com/intl/en-US/health/about/>. Acesso em: 01 dez. 2008.

7. MICROSOFT. Microsoft HealthVault. **HealthVault Overview**, 2008. Disponivel em: <https://account.healthvault.com/help.aspx?topicid=HelpDirectory&rmproc=true>. Acesso em: 01 dez. 2008.

8. PETERS, K. et al. Google Health vs. Microsoft HealthVault WhitePaper. **UserCentric**, 2009. Disponivel em: <http://www.usercentric.com/publications/2009/02/02/google-health-vs-microsoft-healthvault-consumers-compare-online-personal-hea>. Acesso em: 12 mar. 2009.

9. CARTIER, J. C. **Emergency medical information device**. US 2008/0059236 Al, 6 Março 2008.

10. BADILINI, F.; ISOLA, L. **Freeware ECG Viewer for the XML FDA Format**. 2ndOpenECG Workshop. Berlin: [s.n.]. 2004. p. 31-35.

11. ACR, NEMA. Digital Imaging and Communications in Medicine. **DICOM**, 2008. Disponivel em: <http://medical.nema.org/dicom/2008>. Acesso em: 03 nov. 2008.

12. FERRANTI, J. M. et al. The clinical Document Architecture and the Continuity of Care Record: A Critical Analysis. **Journal of the American Informatics Association**, v. 13, n. 3, p. 245-252, Junho 2006.

13. COSTA, C.; SILVA, A.; OLIVEIRA, J. L. Current Prespectives on PACS and a Cardiology Case Study. In: _____ **Studies in Computational Intelligence**. Berlin: Springler-Verlag, 2007. p. 79-108.

14. CHEN, T.-J.; AL., E. Quality Degradation in Lossy Wavelet Image Compression. **Journal of Digital Imaging**, v. 16, n. 2, p. 210-215, 19 February 2004.

15. RADIOLOGY DEPARTMENT GENEVE UNIVERSITY. DICOM sample image sets. **Casimage Database**, 2008. Disponivel em: <http://pubimage.hcuge.ch:8080/>. Acesso em: 12 Novembro 2008.

16. WESTERN DIGITAL. External Hard Drives Overview. **Western Digital**, 2008. Disponivel em: <http://www.wdc.com/en/products/index.asp?cat=9>. Acesso em: 10 Dezembro 2008.

17. TEXAS MEMORY SYSTEMS. Texas Memory Systems : Knowlodge Center. **What is a**

Solid State Disk?, 2008. Disponivel em: <http://www.ramsan.com/whatisassd.htm>. Acesso em: 12 Junho 2008.

18. USB IMPLEMENTERS FORUM. USB 3.0. **USB-IF Technical White Papers**, 2009. Disponivel em: <http://www.usb.org/developers/whitepapers/USB_3_0_SuperSpeed_CDR_0p5_whitepaper.pdf>.

19. SAMSUNG SEMICONDUCTORS. Samsung Semiconductors-Products-Flash SSD. **Samsung Semiconductors**, 2008. Disponivel em: <http://www.samsung.com/global/business/semiconductor/products/flash/Products_FlashSSD.html>. Acesso em: 12 Junho 2008.

20. KUSNETZKY, D. Virtualization is More than Virtual Machine Software. **Kusnetzky Group**, 2007. Disponivel em: <http://www.kusnetzky.net/publications/ImpactPapers/20070829_Virtualization_is_more_than_VM.pdf>.

21. SINGH, A. An Introduction to Virtualization. **Kernel Thread Publications**, 2004. Disponivel em: <http://www.kernelthread.com/publications/virtualization/>. Acesso em: 23 Março 2009.

22. VMWARE. VMware Workstation - Run Multiple OS on Virtual Machines. **VMware Products**, 2009. Disponivel em: <http://www.vmware.com/products/workstation/>. Acesso em: 23 Março 2009.

23. VDI WORKS. How to build an Application Virtualization Framework. **VDI Works:** Virtual Desktop Management, 2008. Disponivel em: <http://vdiworks.com/wp/?p=15>. Acesso em: 10 Maio 2009.

24. MICROSOFT CORPORATION. SoftGrid - Application Virtualization. **Microsoft SGAV White Papers**, 2007. Disponivel em: <http://download.microsoft.com/download/6/4/f/64f5dc66-832a-4df3-baf4-

3b4e7fb9e500/SGAV%20White%20Paper%20-The%20Next%20Frontier.pdf>.

25. CITRIX SYSTEMS. Products: Citrix XenApp. **Citrix Systems**, 2009. Disponivel em: <http://www.citrix.com/english/ps2/products/product.asp?contentid=186>. Acesso em: 12 Janeiro 2009.

26. PORTABLEAPPS.COM. PortableApps.com Suite. **PortableApps.com - Portable Software for USB devices**, 2009. Disponivel em: <http://portableapps.com/suite>.

27. VMWARE. VMware ThinApp - Agentless Application Virtualization Overview. **VMware White Papers**, 2008. Disponivel em: <http://www.vmware.com/files/pdf/thinapp_intro_whitepaper.pdf>.

28. VMWARE. VWware ThinApp. **VMware - Products**, 2009. Disponivel em: <http://www.vmware.com/products/thinapp/howtobuy.html>. Acesso em: 11 Março 2009.

29. NET APLICATIONS. Operating System Market Share. **Global Market Share Statistics**, 2009. Disponivel em: <http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=8>. Acesso em: 18 Julho 2009.

30. DEDASYS. Programming Language Popularity, 2009. Disponivel em: <http://www.langpop.com/>. Acesso em: 12 Julho 2009.

31. SUN MICROSYSTEMS. The Java Language Environment. **Java Technology White Papers**, 1997. Disponivel em: <http://java.sun.com/docs/white/langenv/Intro.doc1.html#943>.

32. MICROSOFT DEVELOPMENT. Overview of the.NET Framework. **Visual Studio Developer Center**, 2009. Disponivel em: <http://msdn.microsoft.com/en-us/library/a4t23ktk.aspx>. Acesso em: 12 Jul 2009.

33. HALLER, J. T. Java Portable - get your Java to go. **PortableApps**, 2009. Disponivel em: <http://portableapps.com/apps/utilities/java_portable>. Acesso em: 12 agosto 2009.

34. TRUECRYPT FOUNDATION. Free Open-Source Disk Encryption Software. **TrueCrypt**, 2009. Disponivel em: <http://www.truecrypt.org/docs/>. Acesso em: 21 Maio 2009.

35. OLZAK, T. W. Evaluation of TrueCrypt as a Mobile Data Encryption Solution. **Erudio Security**, 2008.

36. WINZIP COMPUTING. AES Encryption Information: Encryption Specification. **WinZip**, 2009. Disponivel em: <http://www.winzip.com/aes_info.htm>. Acesso em: 12 maio 2009.

37. DEAN, S. FreeOTFE : About. **FreeOTFE**, 2009. Disponivel em: <http://www.freeotfe.org/index.html>. Acesso em: 12 Março 2009.

38. ARTPOL SOFTWARE. The ZipArchive Library. **Artpol Software**, 2009. Disponivel em: <http://www.artpol-software.com/ZipArchive/Default.aspx>. Acesso em: 23 Maio 2009.

39. HWACI - APPLIED SOFTWARE RESEARCH. SQLite Home page. **SQLite**, 2009. Disponivel em: <http://www.sqlite.org/index.html>. Acesso em: 27 Maio 2009.

40. SQLITE. SQLite Is Self-Contained. **SQLite Specifications**, 2009. Disponivel em: <http://www.sqlite.org/selfcontained.html>. Acesso em: 19 Maio 2009.

41. WILLE, C. Storing Passwords - Done Right! **ASPheute**, 1 Maio 2004. Disponivel em: <http://www.aspheute.com/english/20040105.asp>. Acesso em: 21 Julho 2009.

42. HIPP, WYRICK & COMPANY, INC. SQLite. **The SQLite Encryption Extension (SEE)**, 2008. Disponivel em: <http://www.hwaci.com/sw/sqlite/see.html>. Acesso em: 21 Março 2009.

43. MICROSOFT CORPORATION. Limitações do sistema de ficheiros FAT32 no Windows XP. **Microsoft Support**, 01 Dezembro 2007. Disponivel em: <http://support.microsoft.com/kb/314463>. Acesso em: 01 out. 2009.

44. MICROSOFT CORPORATION. How NTFS Works: Local File Systems. **Microsoft Technet**, 28 Março 2003. Disponivel em: <http://technet.microsoft.com/en-us/library/cc781134%28WS.10%29.aspx>. Acesso em: 1 out. 2009.

45. JPEG COMMITTEE. JPEG 2000 - Compression Standards. **JPEG 200**, 2099. Disponivel em: <http://www.jpeg.org/jpeg2000/index.html>. Acesso em: 12 Maio 2009.

46. EYSENBACH, G. Medicine 2.0: Social Networking, Collaboration, Participation, Apomediation, and Openness. **Journal of Medical Internet Research**, Agosto 2008. Acesso em: 02 dez. 2008.

47. JADAD, A. R.; DELAMOTHE, T. What next for electronic communication and health care? **British Medical Journal**, London, v. 328, p. 1143-1144, Maio 2004.

48. MCDONALD, C. J. The Barriers to Electronic Medical Record System and How to Overcome Them. **Journal of the American Informatics Association**, Indiana, v. IV, n. 3, p. 213-221, May 1997.

49. UCKERT, F. K.; PROKOSCH, H.-U. **Implementing Security and Acess Control Mechanisms For An Electronic Healthcare Record**. AMIA Annual Symposium. Germany: AMIA Press. 2002. p. 825-829.